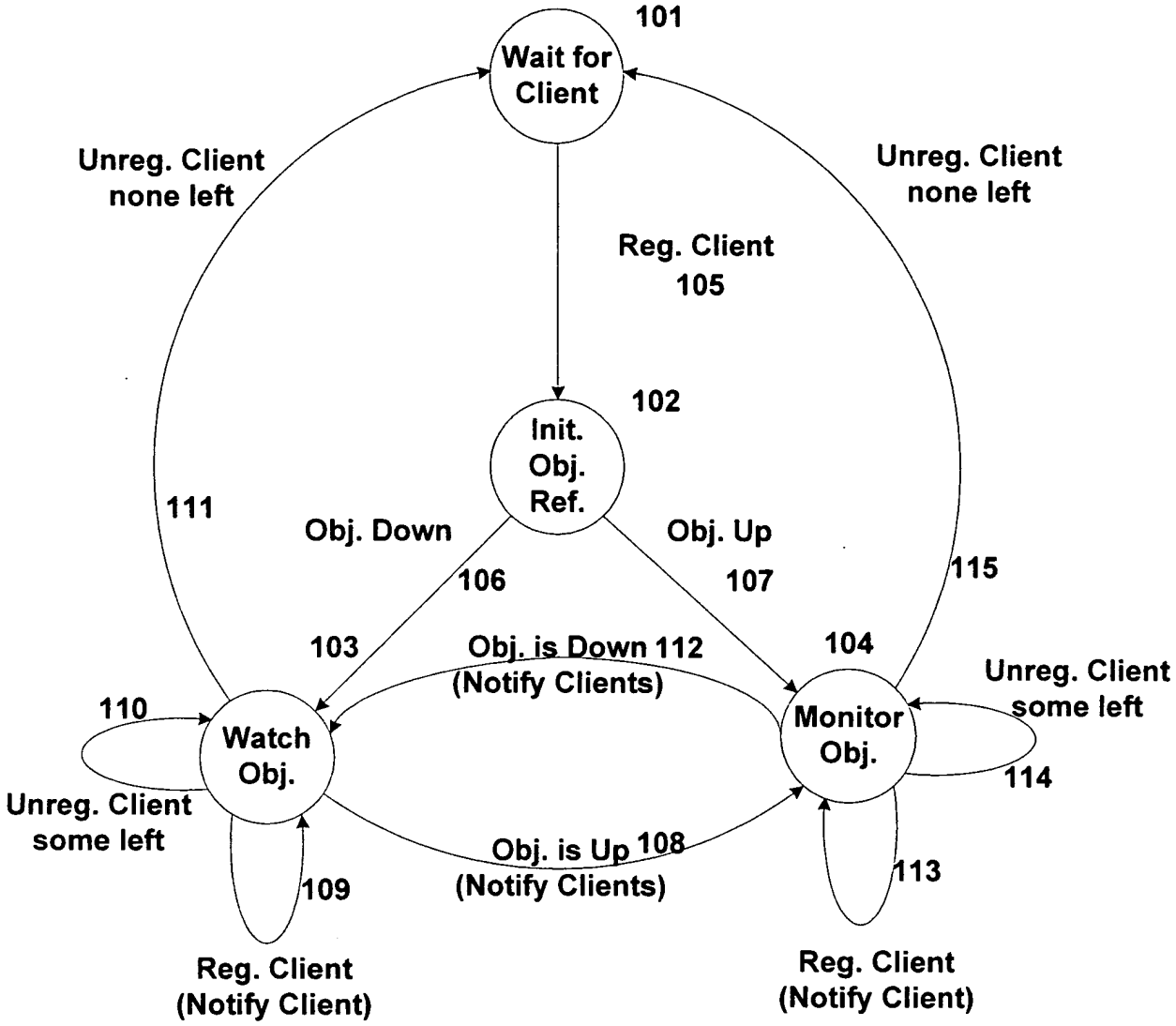
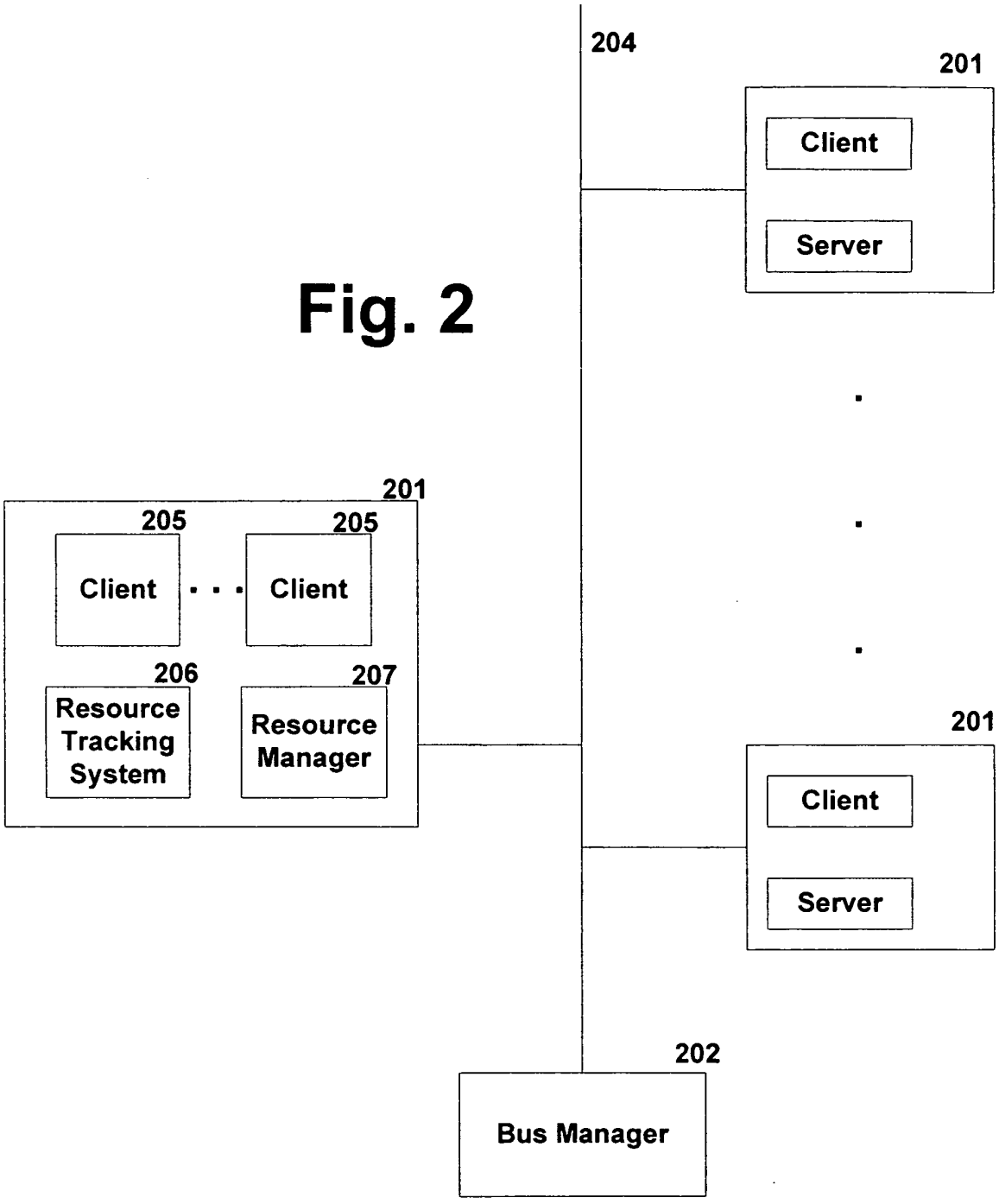

101
Wait for Client
Unreg. Client none left
Unreg. Client none left
Reg. Client 105
102
Init. Obj. Ref.
111
Obj. Down
Obj. Up
106
107
115
103
Obj. is Down 112 (Notify Clients)
104
Unreg. Client some left
110
Watch Obj.
Monitor Obj.
114
Unreg. Client some left
Obj. is Up 108 (Notify Clients)
109
113
Reg. Client (Notify Client)
Reg. Client (Notify Client)

Fig. 1

Fig. 2


204
201
Client
Server
201
205
205
Client
Client
206
207
Resource Tracking System
Resource Manager
201
Client
Server
202
Bus Manager

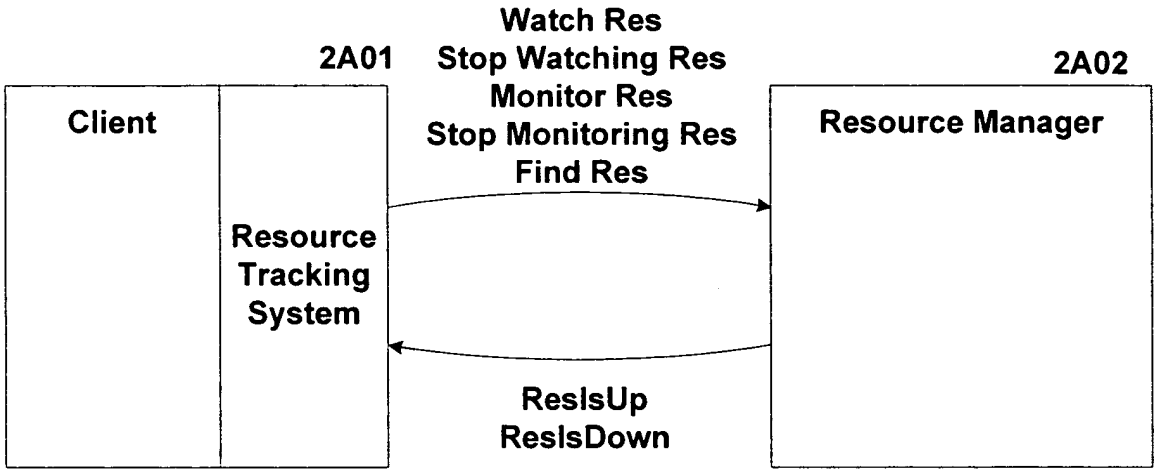

Watch Res
2A01
Stop Watching Res
2A02
Monitor Res
Client
Resource Manager
Stop Monitoring Res
Find Res
Resource
Tracking
System
ResIsUp
ResIsDown

Fig. 2A

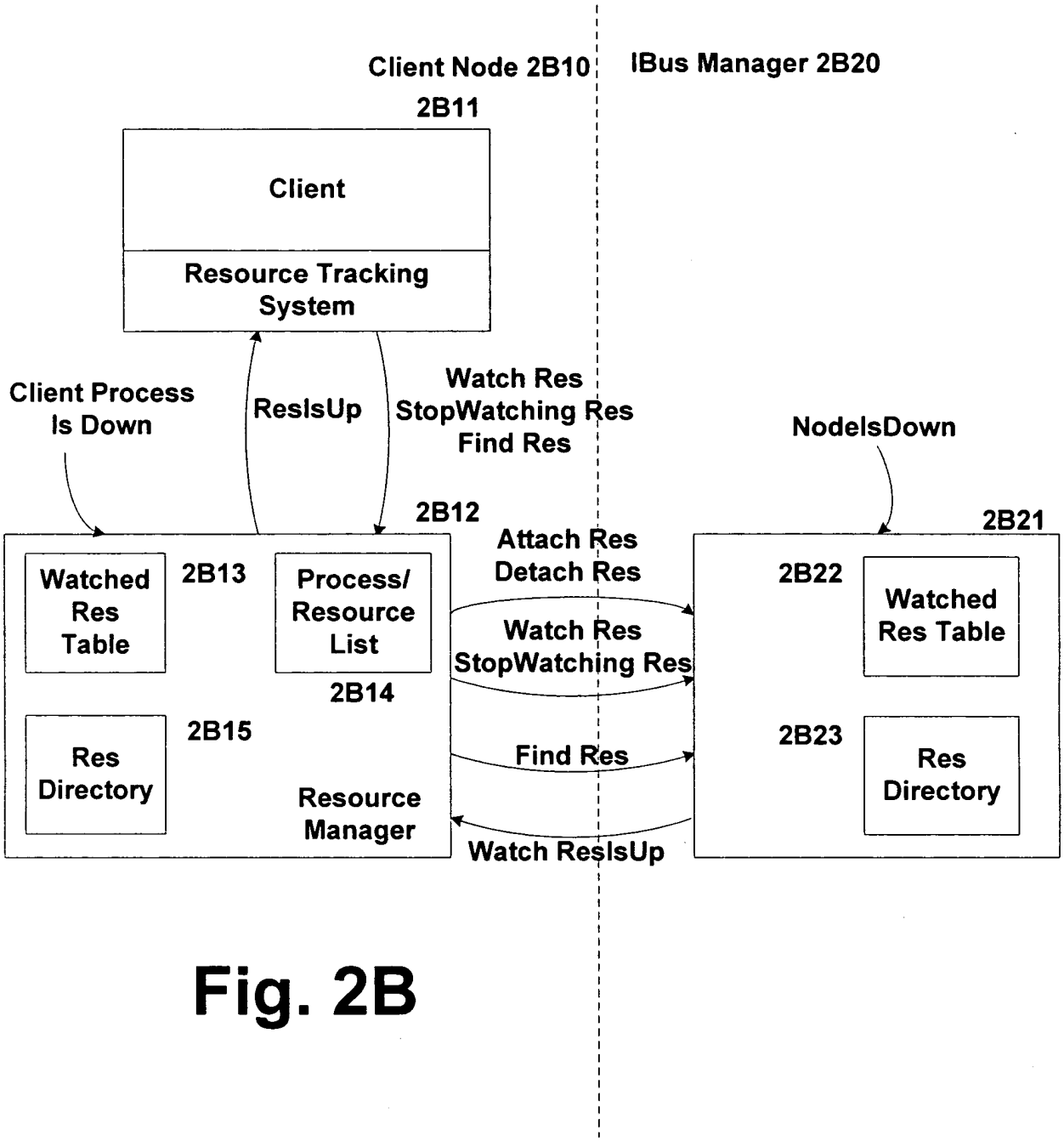

Client Node 2B10
IBus Manager 2B20
2B11
Client
Resource Tracking System
Client Process Is Down
ResIsUp
Watch Res
StopWatching Res
Find Res
NodeIsDown
2B12
2B21
Attach Res
Detach Res
Watched Res Table
2B13
Process/ Resource List
2B14
2B22
Watched Res Table
Watch Res
StopWatching Res
Res Directory
2B15
Find Res
2B23
Res Directory
Resource Manager
Watch ResIsUp

Fig. 2B

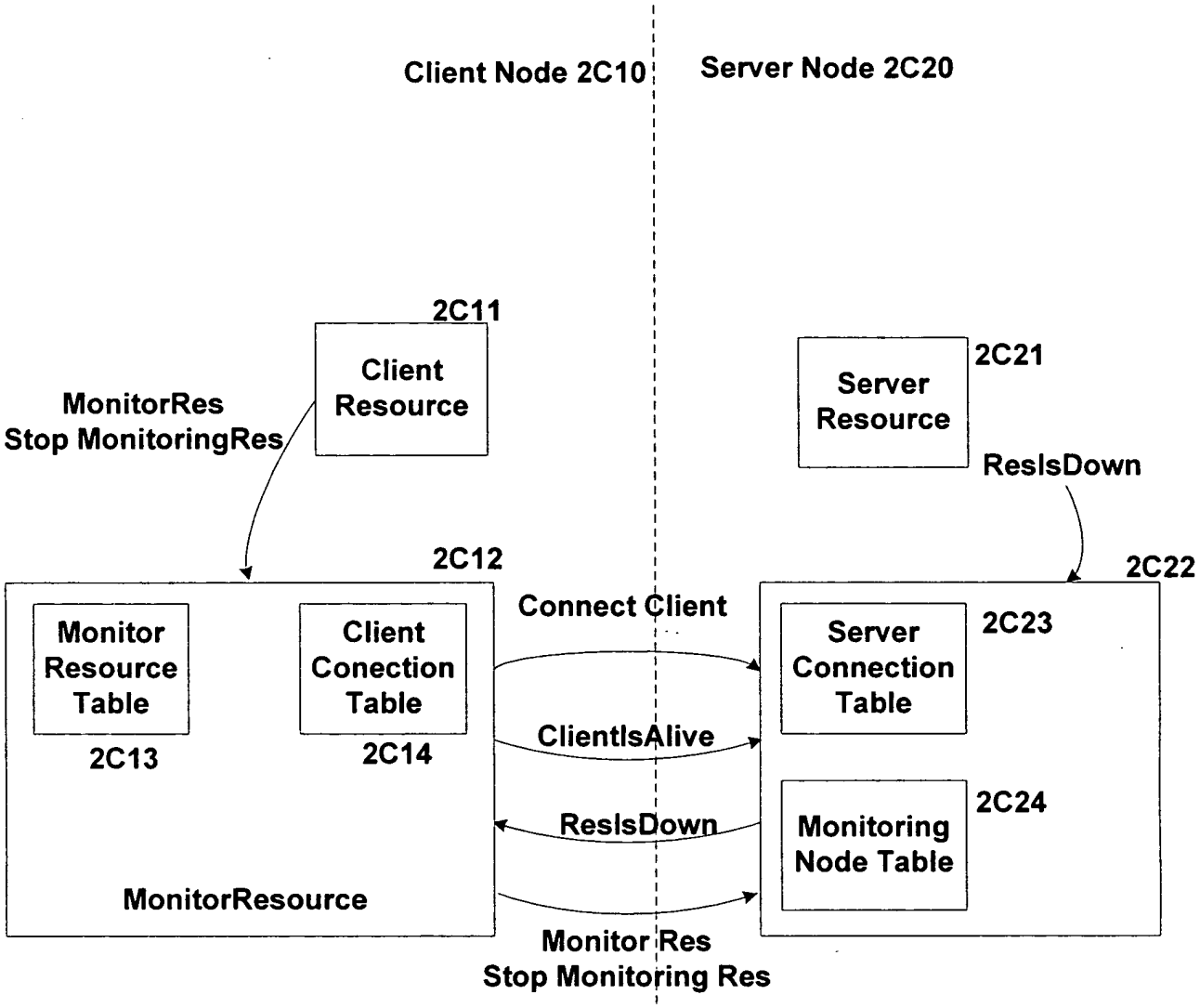

Client Node 2C10
Server Node 2C20
2C11
Client Resource
MonitorRes
Stop MonitoringRes
2C21
Server Resource
ResIsDown
2C12
2C22
Monitor Resource Table
2C13
Client Conection Table
2C14
Connect Client
ClientIsAlive
ResIsDown
MonitorResource
Server Connection Table
2C23
Monitoring Node Table
2C24
Monitor Res
Stop Monitoring Res

Fig. 2C

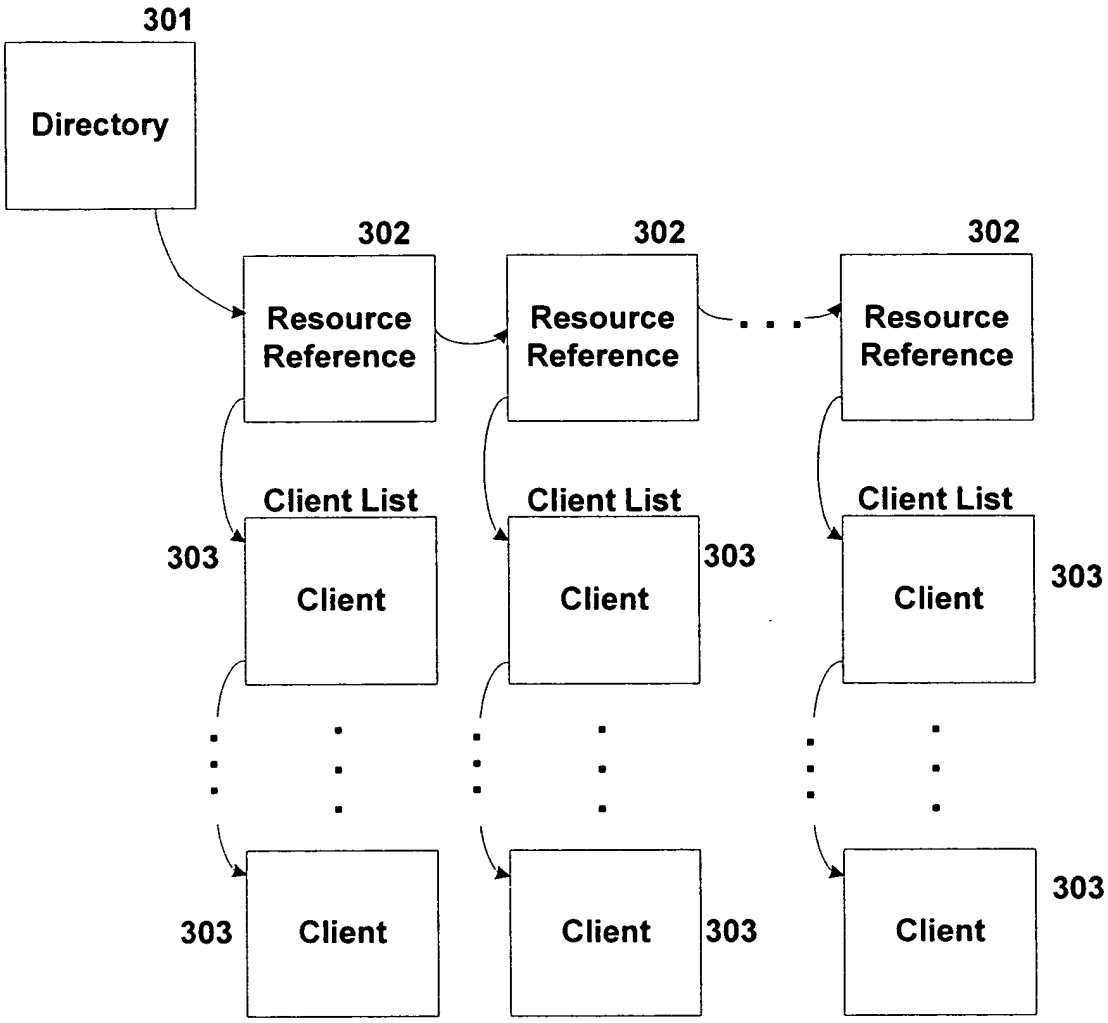

301
Directory
302
Resource Reference
302
Resource Reference
302
Resource Reference
Client List
303
Client
Client List
303
Client
Client List
303
Client
303
Client
303
Client
303
Client

Fig. 3

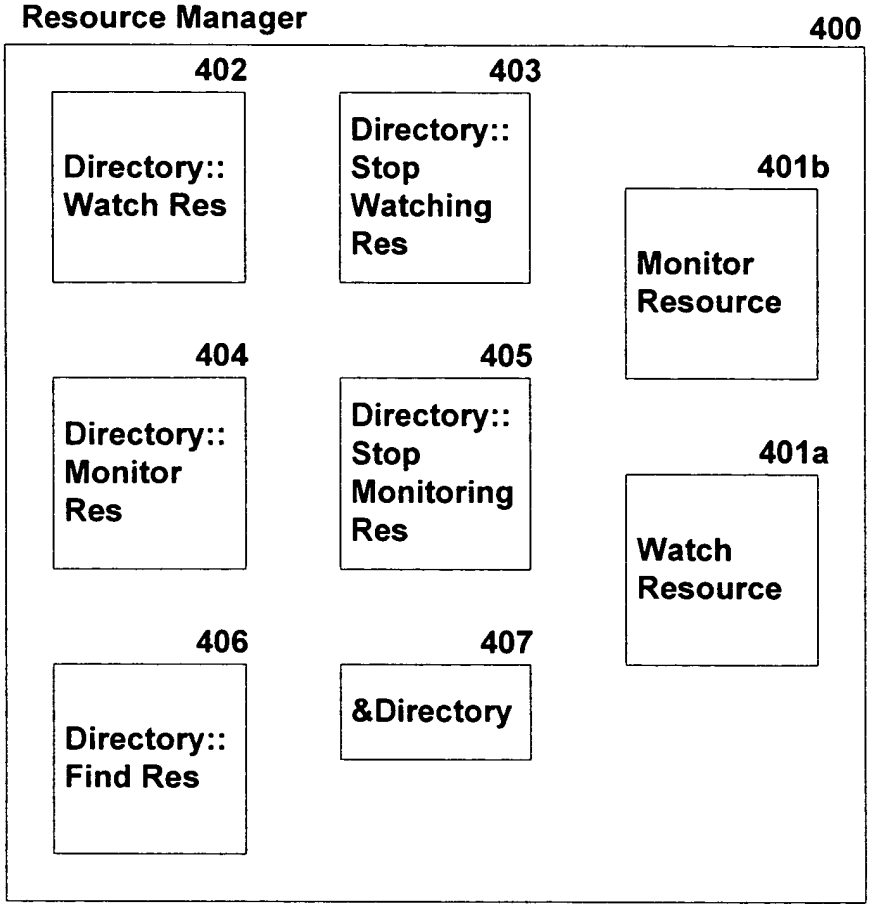

Resource Manager
400
402
Directory:: Watch Res
403
Directory:: Stop Watching Res
401b
Monitor Resource
404
Directory:: Monitor Res
405
Directory:: Stop Monitoring Res
401a
Watch Resource
406
Directory:: Find Res
407
&Directory

Fig. 4

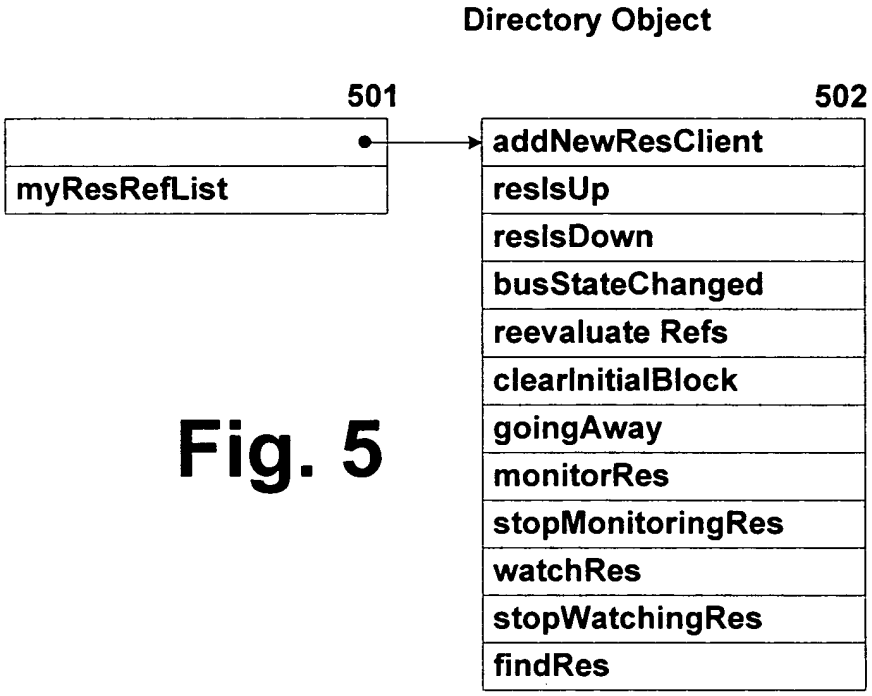

Directory Object
501
502
myResRefList
addNewResClient
resIsUp
resIsDown
busStateChanged
reevaluate Refs
clearInitialBlock
goingAway
monitorRes
stopMonitoringRes
watchRes
stopWatchingRes
findRes

Fig. 5

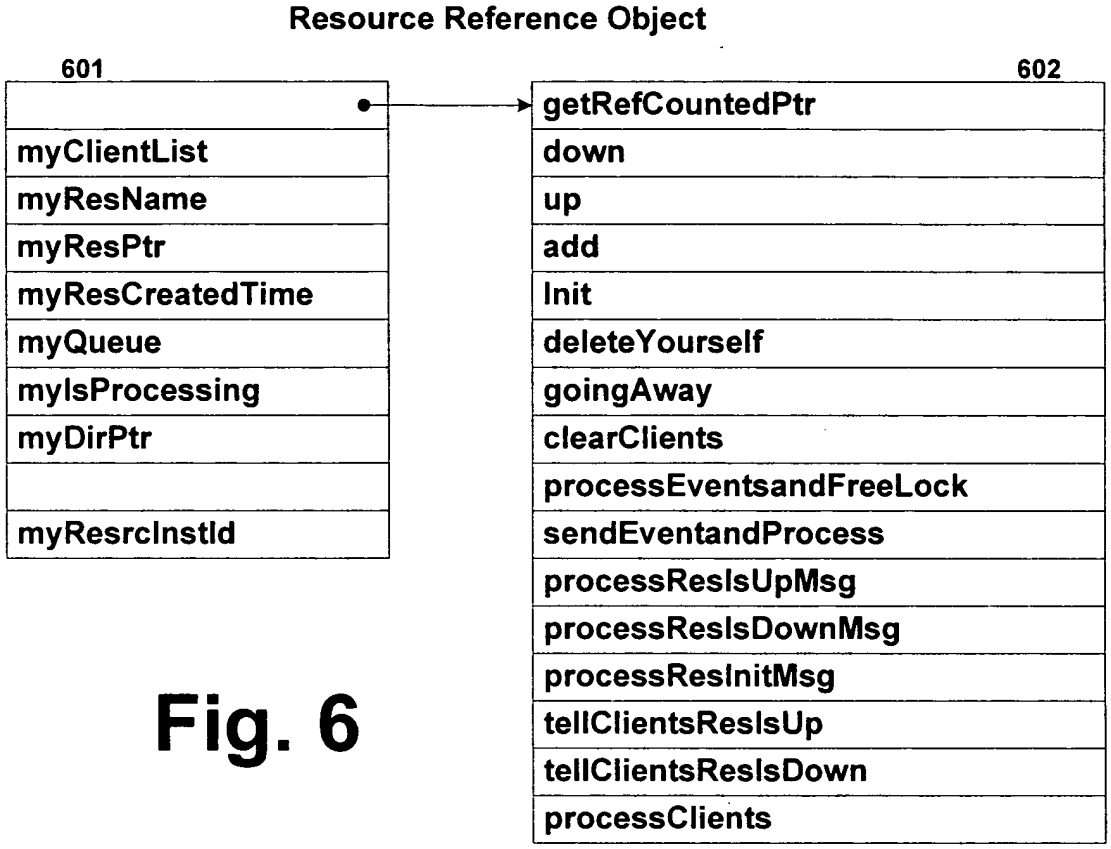

Resource Reference Object
601
602
myClientList
myResName
myResPtr
myResCreatedTime
myQueue
myIsProcessing
myDirPtr
myResrcInstId
getRefCountedPtr
down
up
add
Init
deleteYourself
goingAway
clearClients
processEventsandFreeLock
sendEventandProcess
processResIsUpMsg
processResIsDownMsg
processResInitMsg
tellClientsResIsUp
tellClientsResIsDown
processClients

Fig. 6

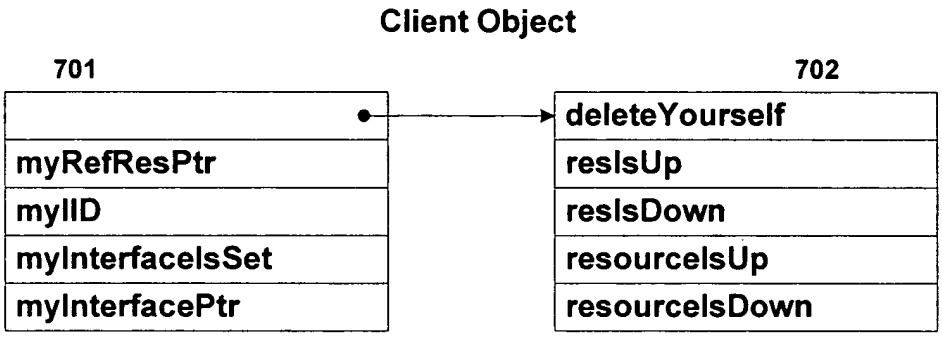

Client Object
701
702
myRefResPtr
myIID
myInterfaceIsSet
myInterfacePtr
deleteYourself
resIsUp
resIsDown
resourceIsUp
resourceIsDown

Fig. 7

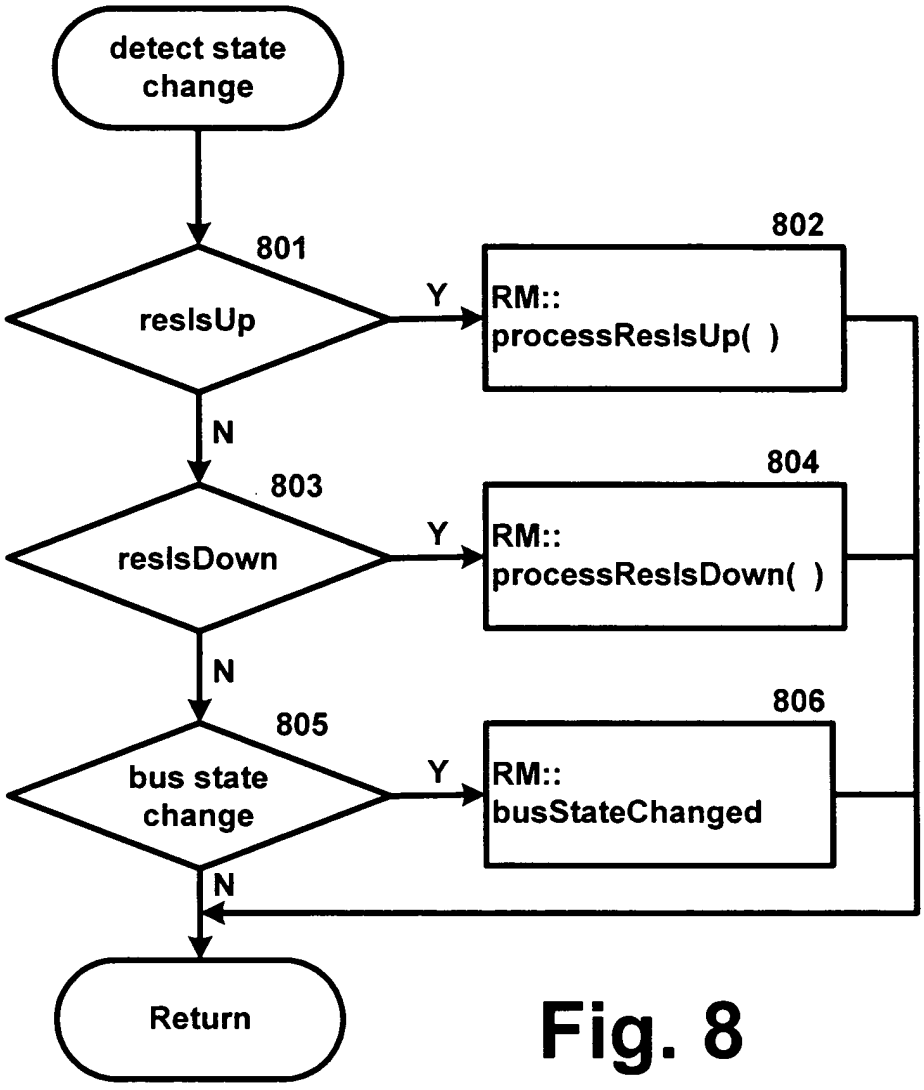

detect state change
801
resIsUp
Y
802
RM::
processResIsUp( )
N
803
resIsDown
Y
804
RM::
processResIsDown( )
N
805
bus state change
Y
806
RM::
busStateChanged
N
Return

Fig. 8

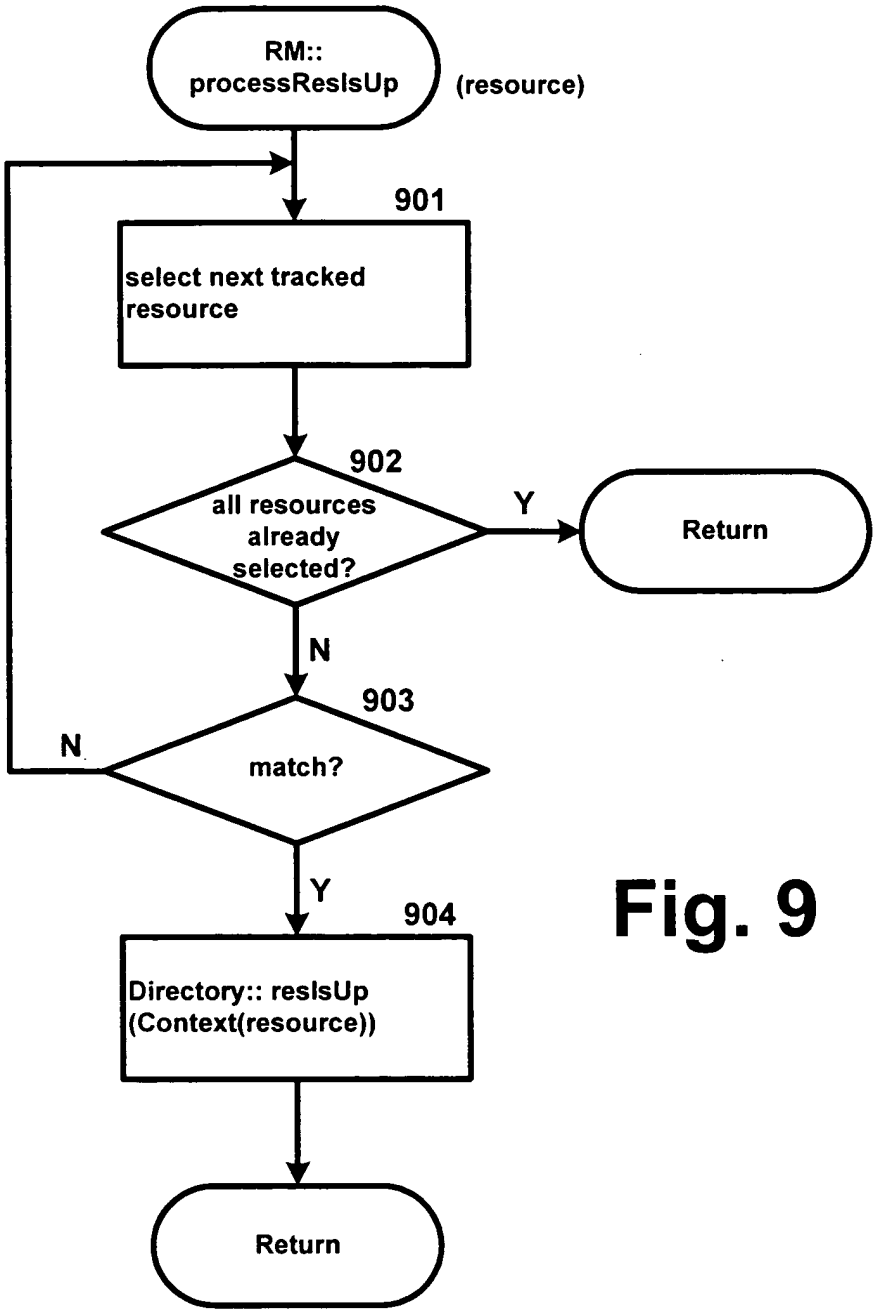

RM::
processResIsUp
(resource)
901
select next tracked resource
902
all resources already selected?
Y
Return
N
903
match?
N
Y
904
Directory:: resIsUp
(Context(resource))
Return

Fig. 9

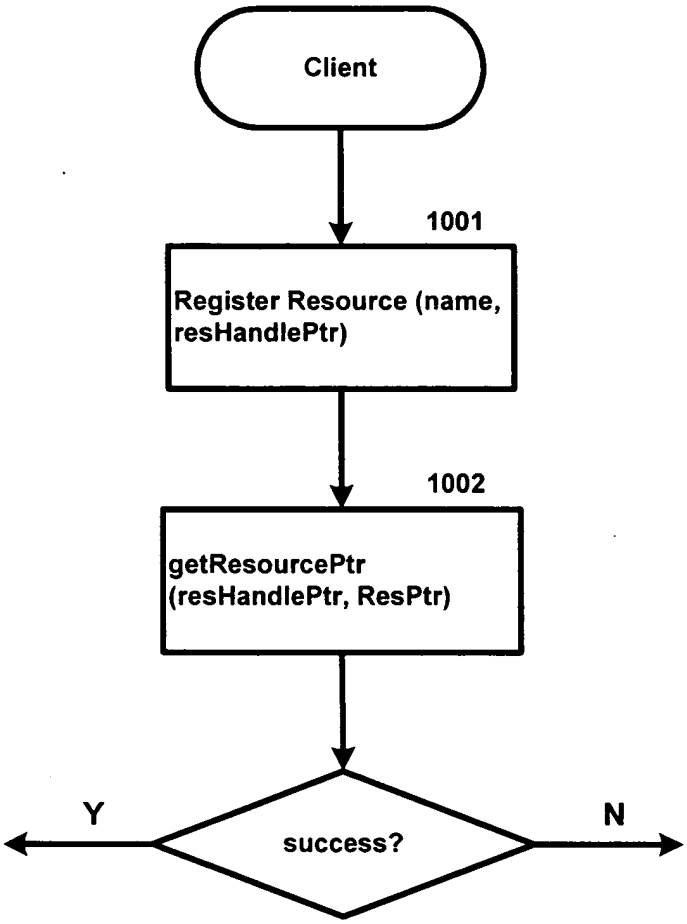

Client
1001
Register Resource (name, resHandlePtr)
1002
getResourcePtr (resHandlePtr, ResPtr)
success?
Y
N

Fig. 10

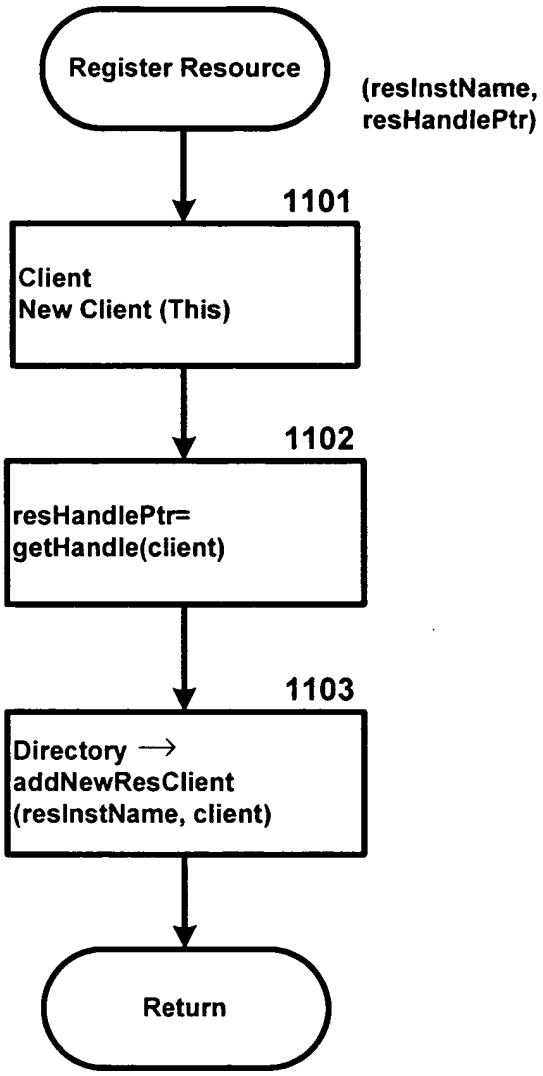

Register Resource
(resInstName, resHandlePtr)
1101
Client
New Client (This)
1102
resHandlePtr=
getHandle(client)
1103
Directory →
addNewResClient
(resInstName, client)
Return

Fig. 11

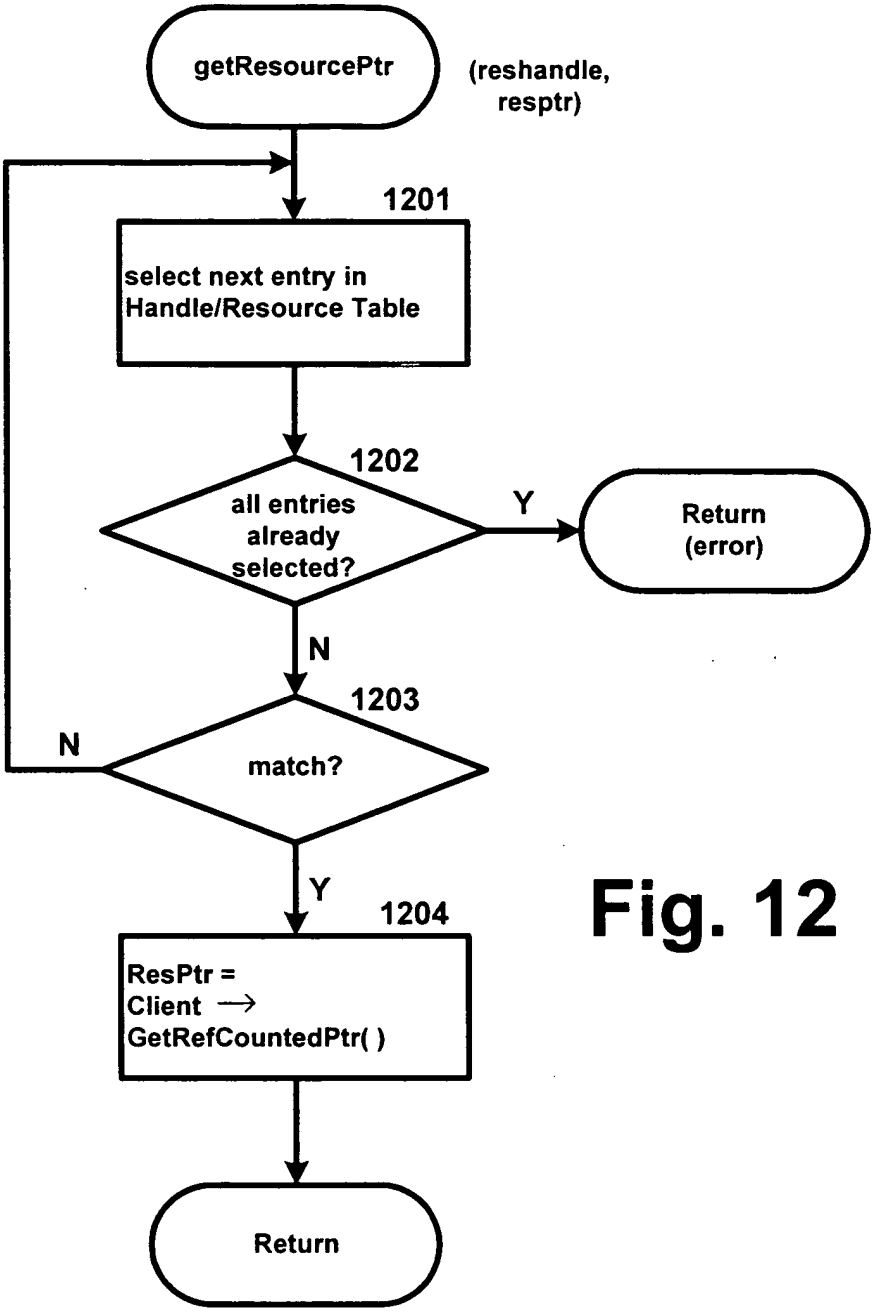

getResourcePtr
(reshandle, resptr)
1201
select next entry in Handle/Resource Table
1202
all entries already selected?
Y
Return (error)
N
1203
match?
N
Y
1204
ResPtr = Client → GetRefCountedPtr( )
Return

Fig. 12

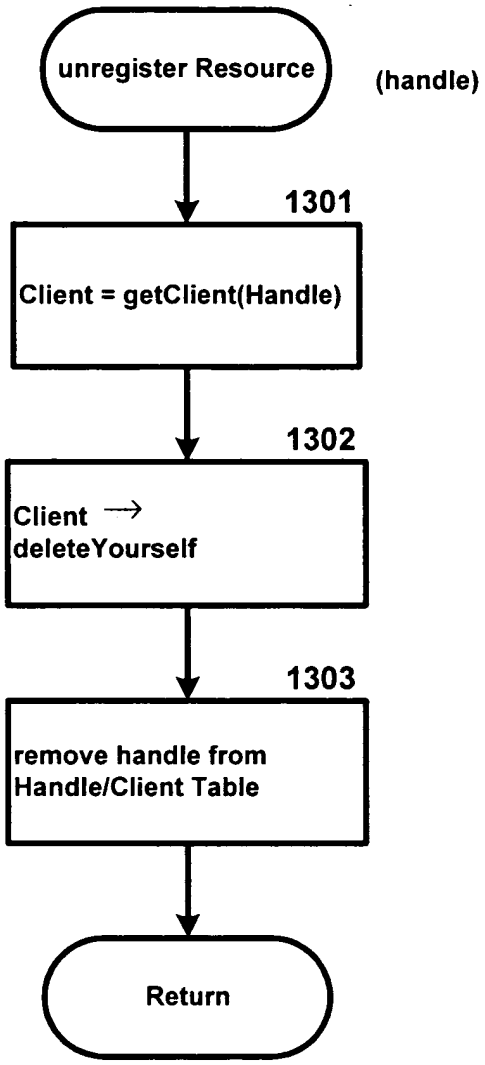

unregister Resource
(handle)
1301
Client = getClient(Handle)
1302
Client → deleteYourself
1303
remove handle from Handle/Client Table
Return

Fig. 13

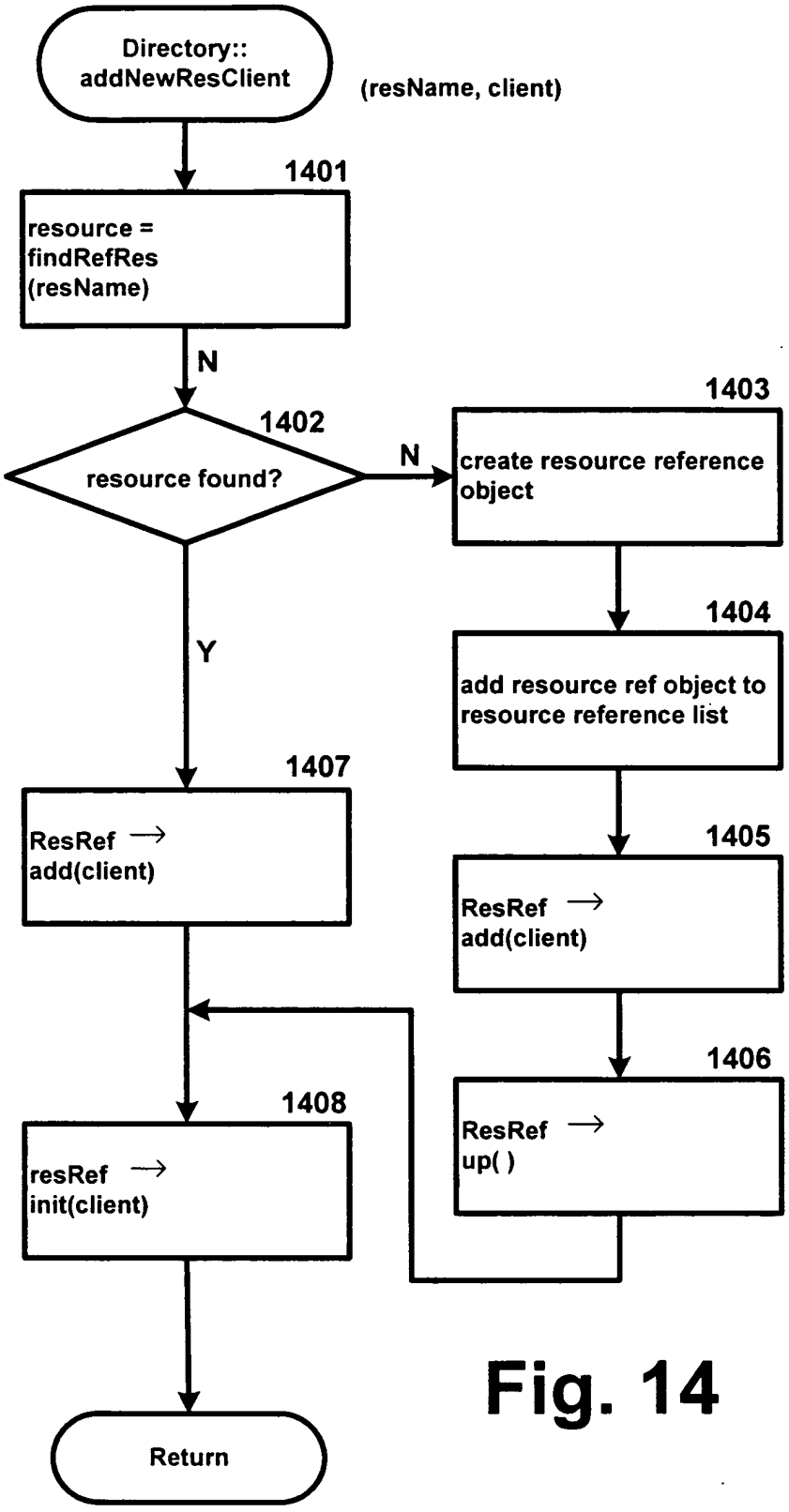

Directory::
addNewResClient
(resName, client)
1401
resource =
findRefRes
(resName)
N
1402
resource found?
N
1403
create resource reference
object
1404
add resource ref object to
resource reference list
Y
1407
ResRef →
add(client)
1405
ResRef →
add(client)
1406
ResRef →
up( )
1408
resRef →
init(client)
Return

Fig. 14

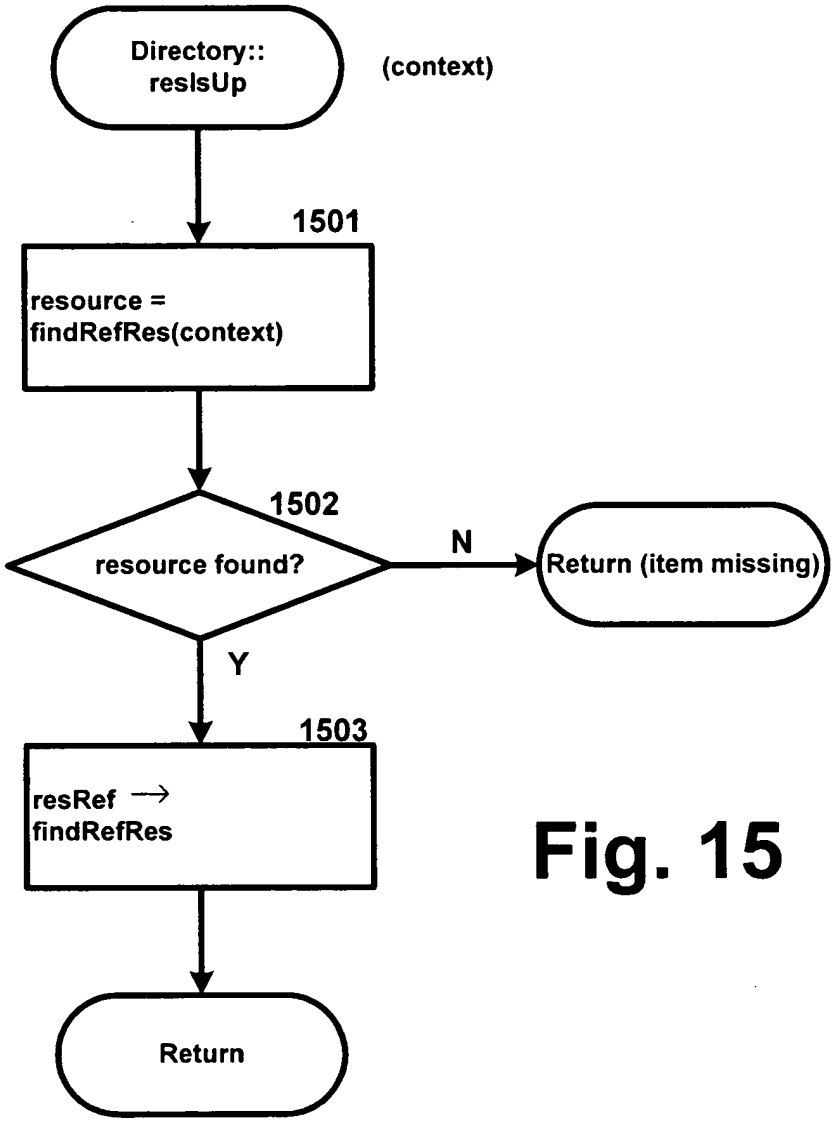

Directory::
resIsUp
(context)
1501
resource =
findRefRes(context)
1502
resource found?
N
Return (item missing)
Y
1503
resRef →
findRefRes
Return

Fig. 15

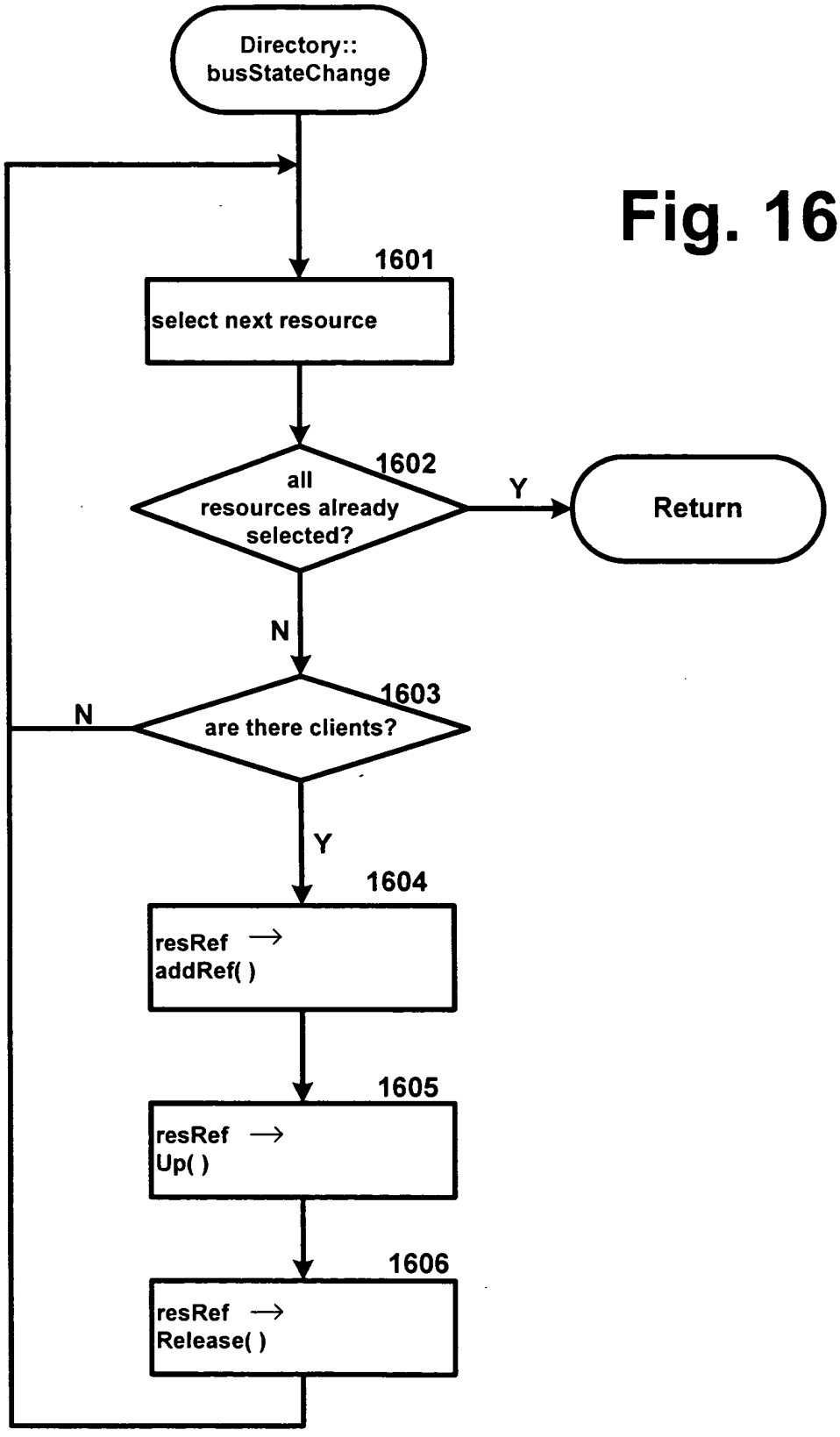

Directory::
busStateChange
1601
select next resource
1602
all
resources already
selected?
Y
Return
N
1603
are there clients?
N
Y
1604
resRef →
addRef( )
1605
resRef →
Up( )
1606
resRef →
Release( )

Fig. 16

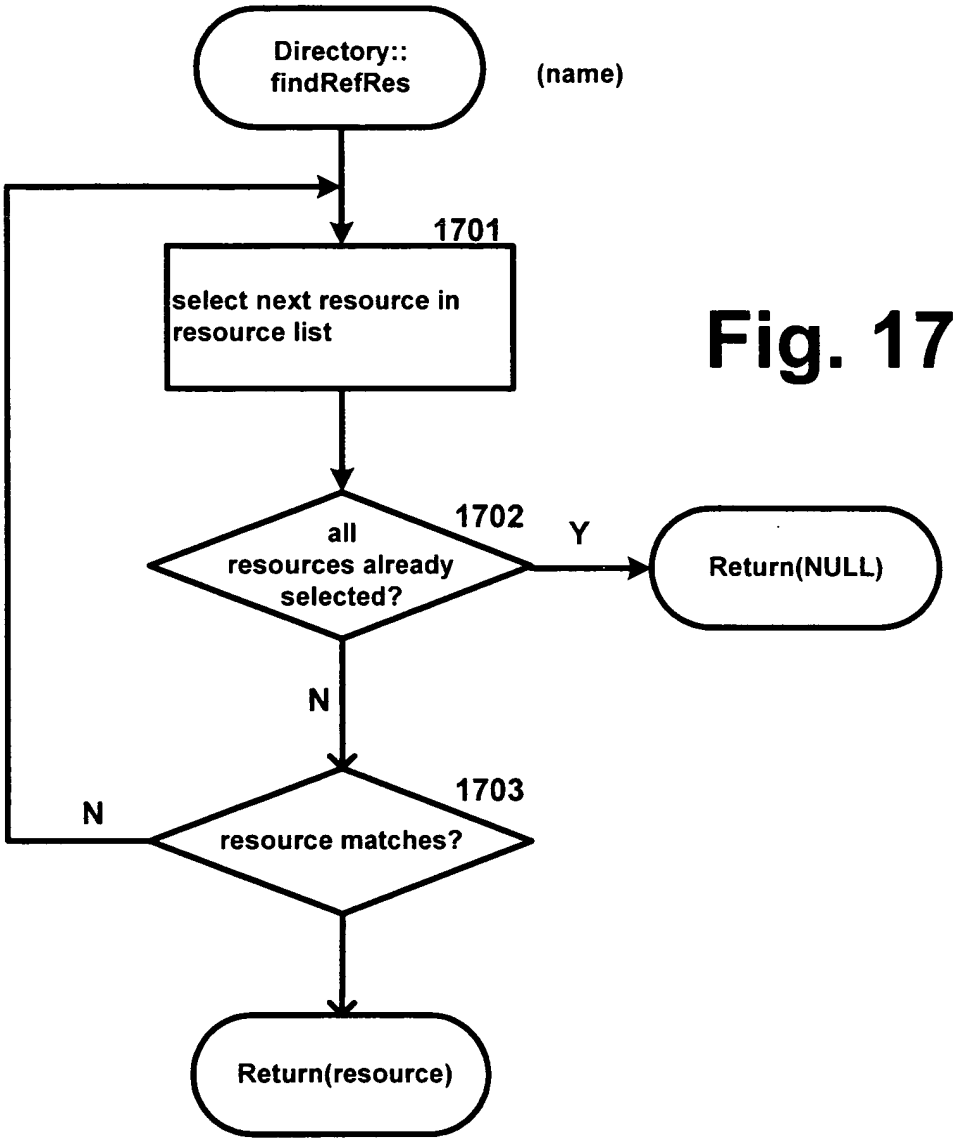

Directory::
findRefRes
(name)
1701
select next resource in
resource list
1702
all
resources already
selected?
Y
Return(NULL)
N
1703
N
resource matches?
Return(resource)

Fig. 17

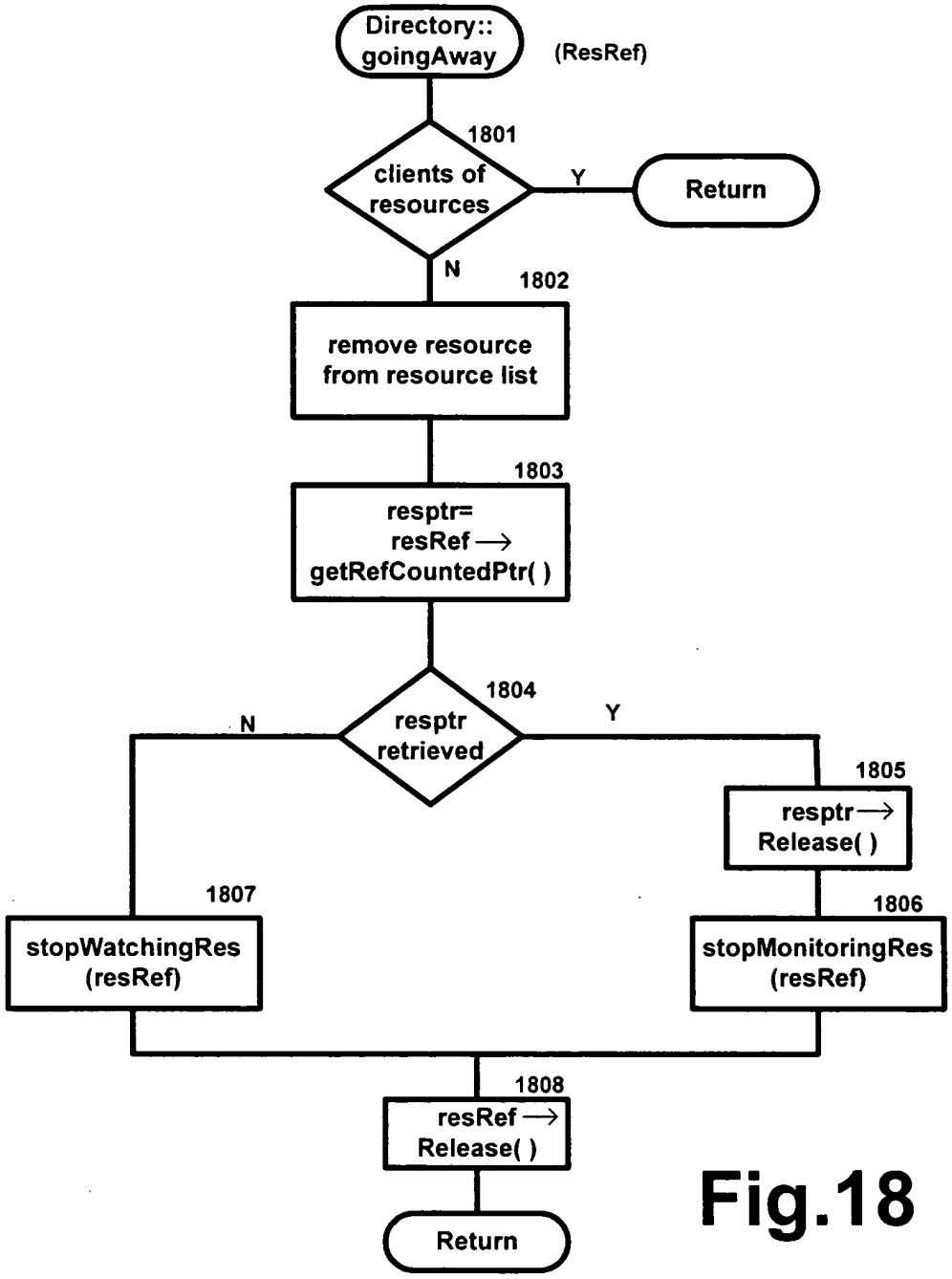

Directory:: goingAway
(ResRef)
1801
clients of resources
Y
Return
N
1802
remove resource from resource list
1803
resptr= resRef → getRefCountedPtr( )
1804
resptr retrieved
N
Y
1805
resptr → Release( )
1807
stopWatchingRes (resRef)
1806
stopMonitoringRes (resRef)
1808
resRef → Release( )
Return

Fig.18

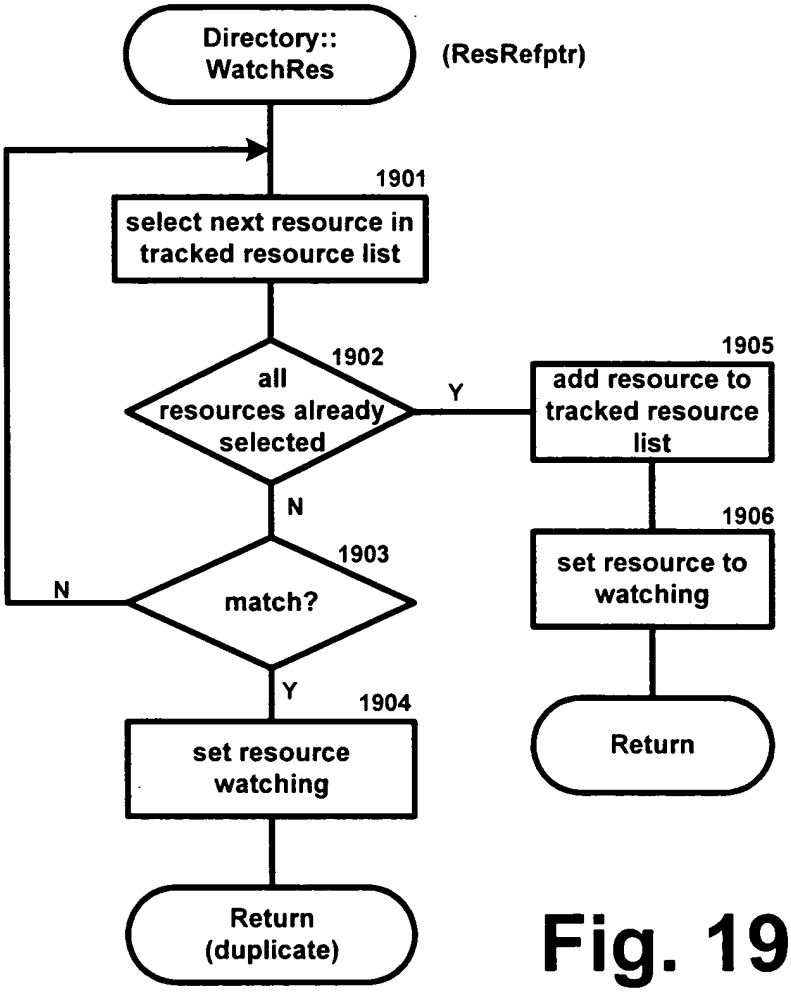

Directory:: WatchRes
(ResRefptr)
1901
select next resource in tracked resource list
1902
all resources already selected
Y
1905
add resource to tracked resource list
N
1906
set resource to watching
1903
match?
N
Y
1904
set resource watching
Return
Return (duplicate)

Fig. 19

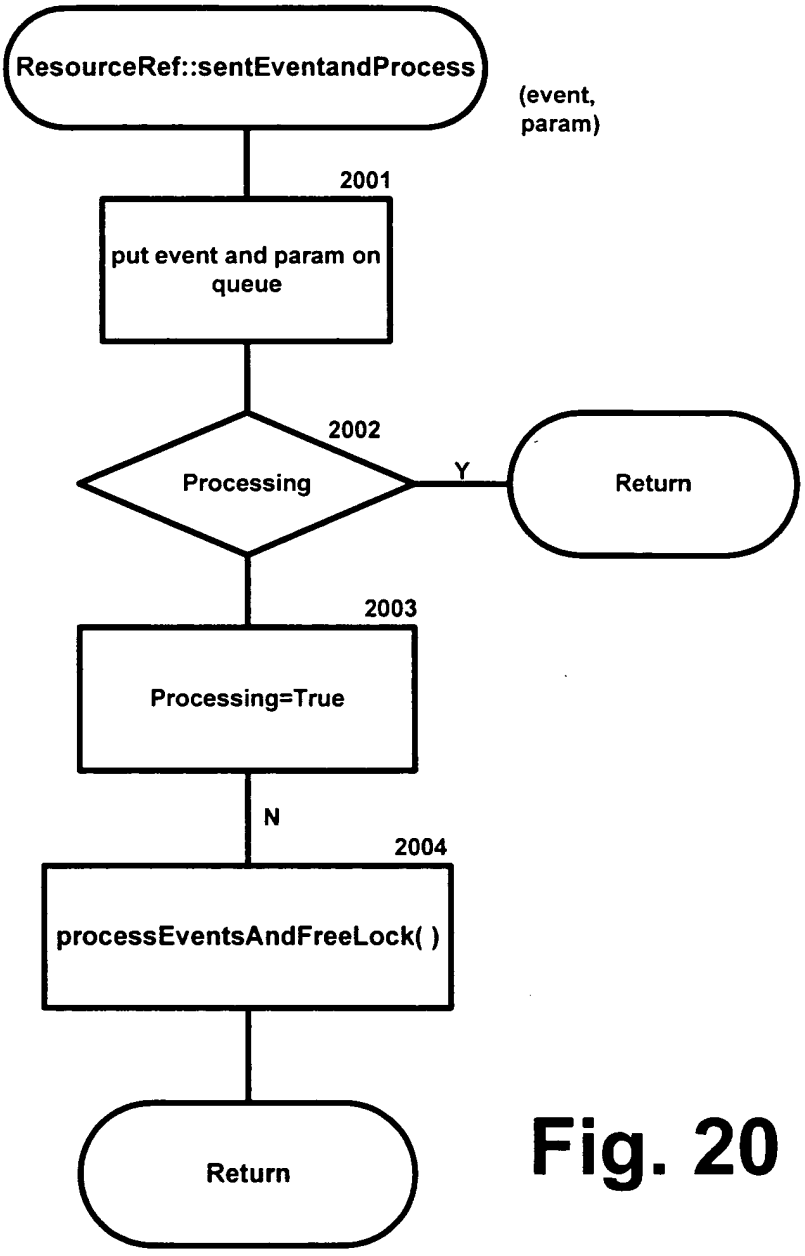

ResourceRef::sentEventandProcess
(event, param)
2001
put event and param on queue
2002
Processing
Y
Return
2003
Processing=True
N
2004
processEventsAndFreeLock( )
Return

Fig. 20

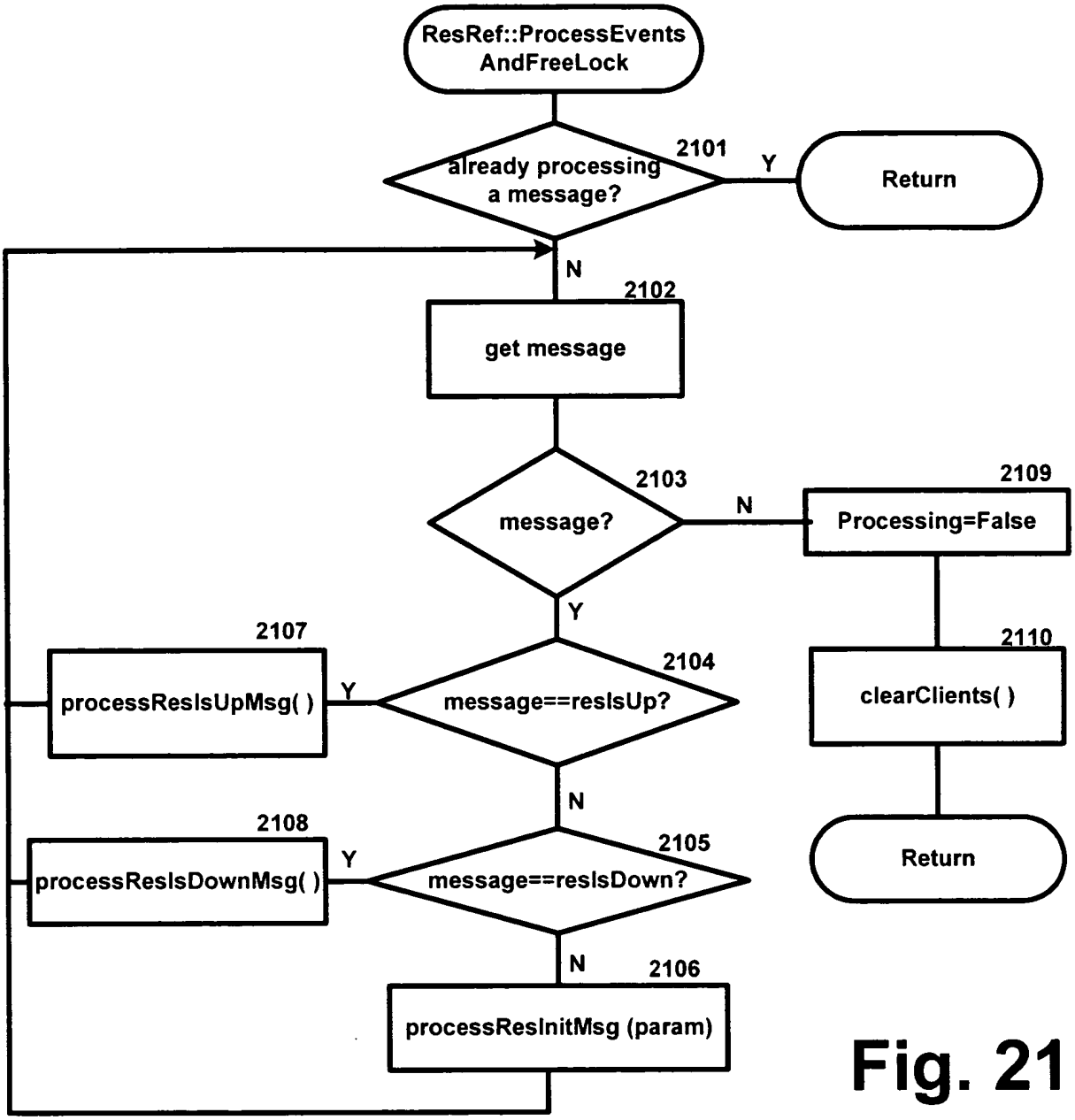

ResRef::ProcessEvents AndFreeLock
2101
already processing a message?
Y
Return
N
2102
get message
2103
message?
N
2109
Processing=False
Y
2107
processResIsUpMsg( )
Y
2104
message==resIsUp?
2110
clearClients( )
N
2108
processResIsDownMsg( )
Y
2105
message==resIsDown?
Return
N
2106
processResInitMsg (param)

Fig. 21

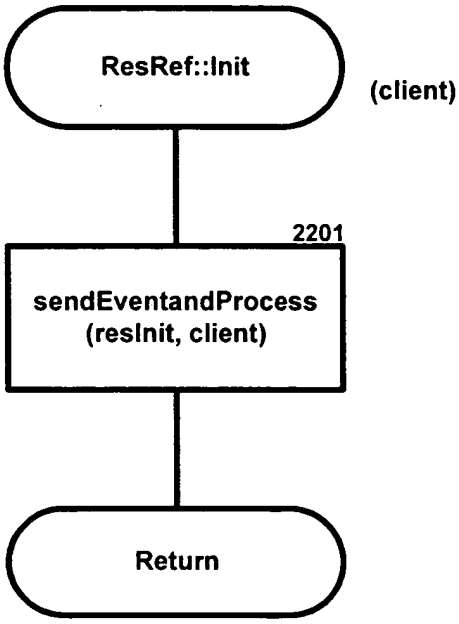

ResRef::Init
(client)
2201
sendEventandProcess
(resInit, client)
Return

Fig. 22

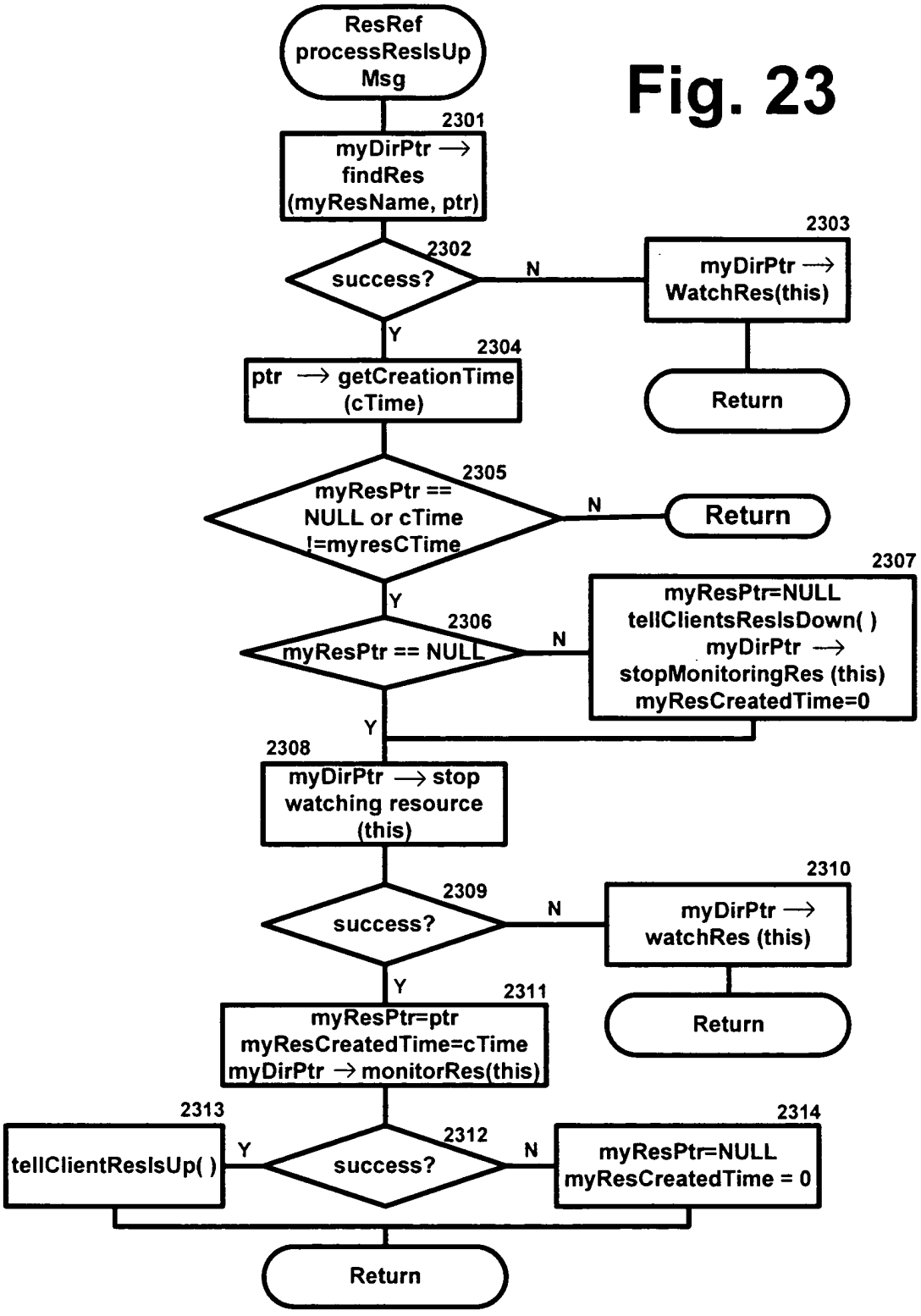

ResRef processResIsUp Msg
2301
myDirPtr → findRes (myResName, ptr)
2302
success?
N
2303
myDirPtr → WatchRes(this)
Return
Y
2304
ptr → getCreationTime (cTime)
2305
myResPtr == NULL or cTime !=myresCTime
N
Return
Y
2307
myResPtr=NULL
tellClientsResIsDown( )
myDirPtr → stopMonitoringRes (this)
myResCreatedTime=0
2306
myResPtr == NULL
N
Y
2308
myDirPtr → stop watching resource (this)
2309
success?
N
2310
myDirPtr → watchRes (this)
Return
Y
2311
myResPtr=ptr
myResCreatedTime=cTime
myDirPtr → monitorRes(this)
2313
tellClientResIsUp( )
Y
2312
success?
N
2314
myResPtr=NULL
myResCreatedTime = 0
Return

Fig. 23

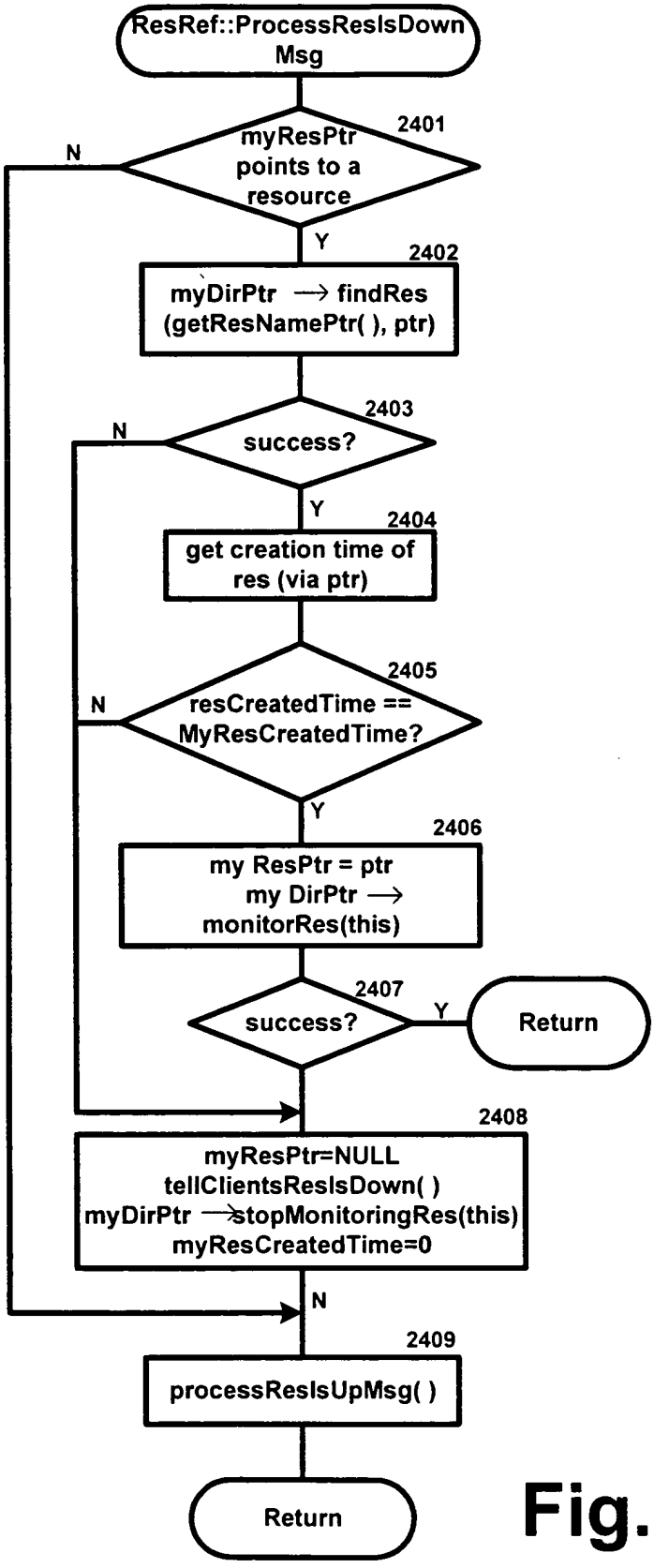

ResRef::ProcessResIsDown Msg
2401
myResPtr points to a resource
N
Y
2402
myDirPtr → findRes (getResNamePtr( ), ptr)
2403
success?
N
Y
2404
get creation time of res (via ptr)
2405
resCreatedTime == MyResCreatedTime?
N
Y
2406
my ResPtr = ptr
my DirPtr → monitorRes(this)
2407
success?
Y
Return
2408
myResPtr=NULL
tellClientsResIsDown( )
myDirPtr →stopMonitoringRes(this)
myResCreatedTime=0
N
2409
processResIsUpMsg( )
Return

Fig. 24

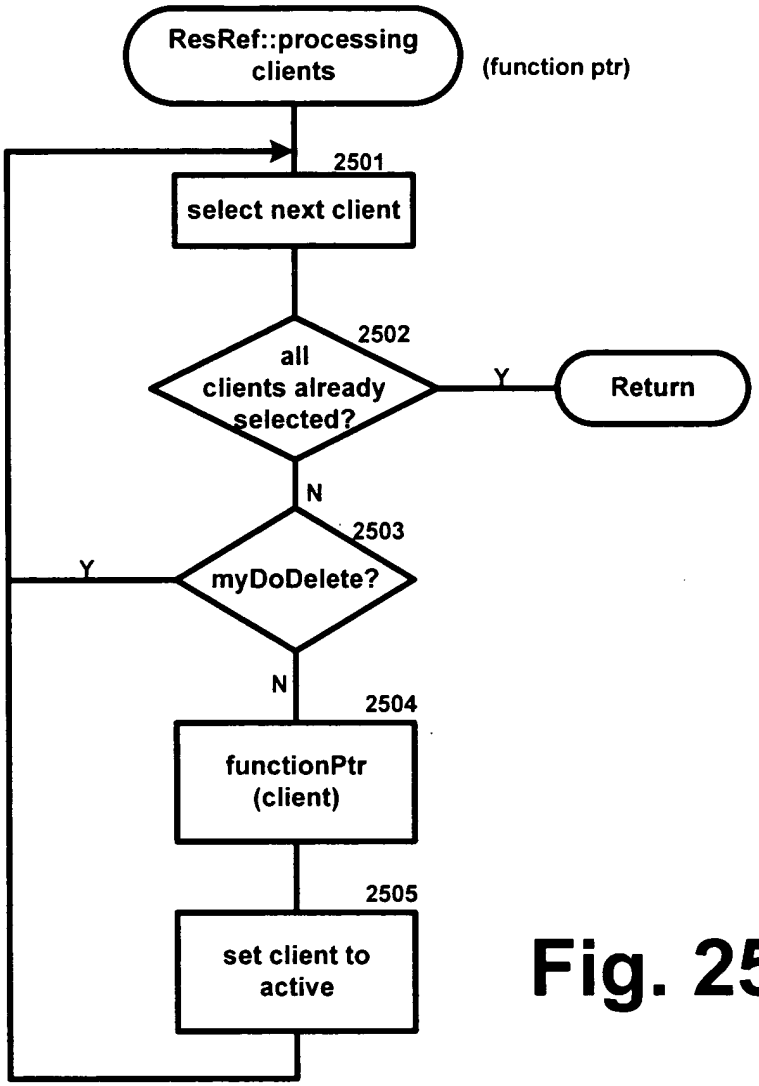

ResRef::processing clients
(function ptr)
2501
select next client
2502
all clients already selected?
Y
Return
N
2503
myDoDelete?
Y
N
2504
functionPtr (client)
2505
set client to active

Fig. 25

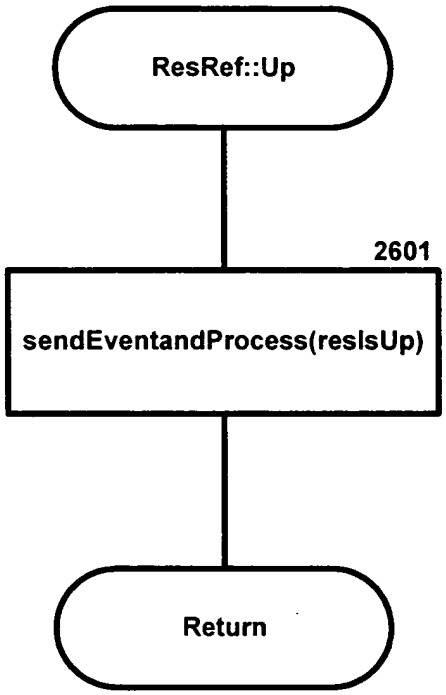


Fig. 26

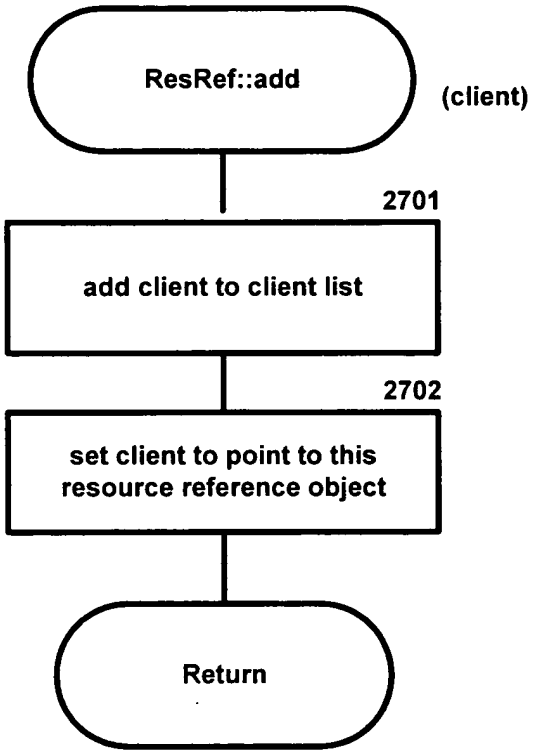

ResRef::add
(client)
2701
add client to client list
2702
set client to point to this resource reference object
Return

Fig. 27

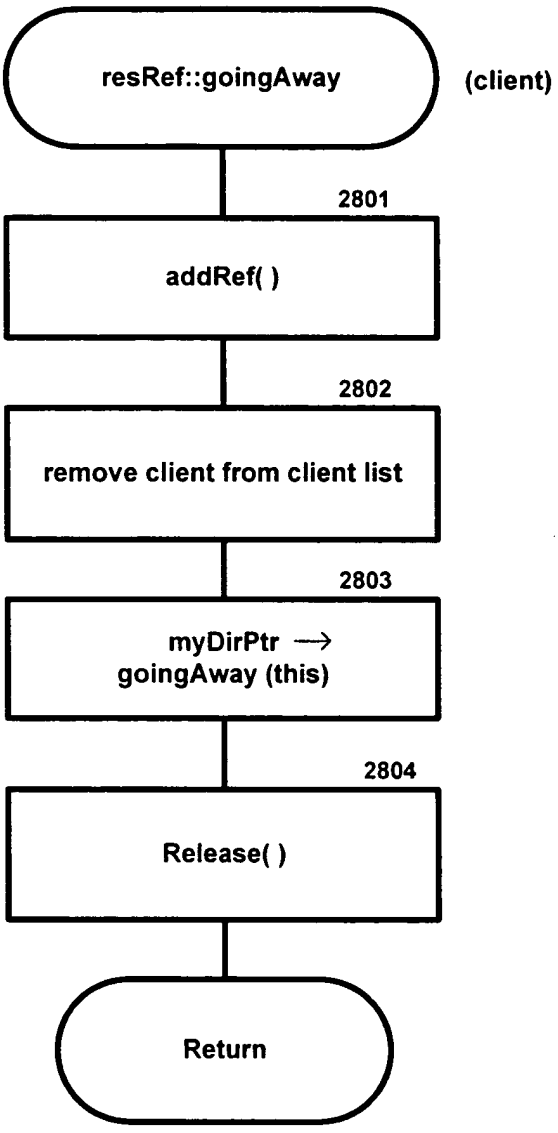

resRef::goingAway
(client)
2801
addRef( )
2802
remove client from client list
2803
myDirPtr → goingAway (this)
2804
Release( )
Return

Fig. 28

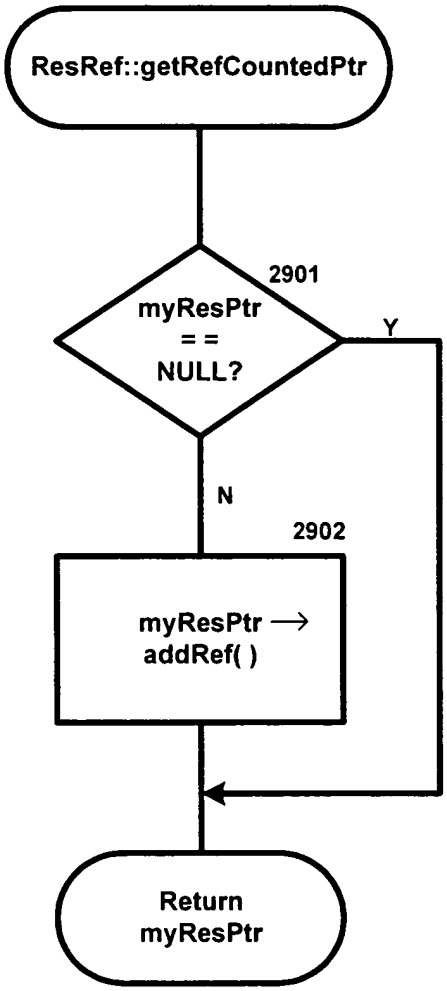

ResRef::getRefCountedPtr
2901
myResPtr
= =
NULL?
Y
N
2902
myResPtr →
addRef( )
Return
myResPtr

Fig. 29

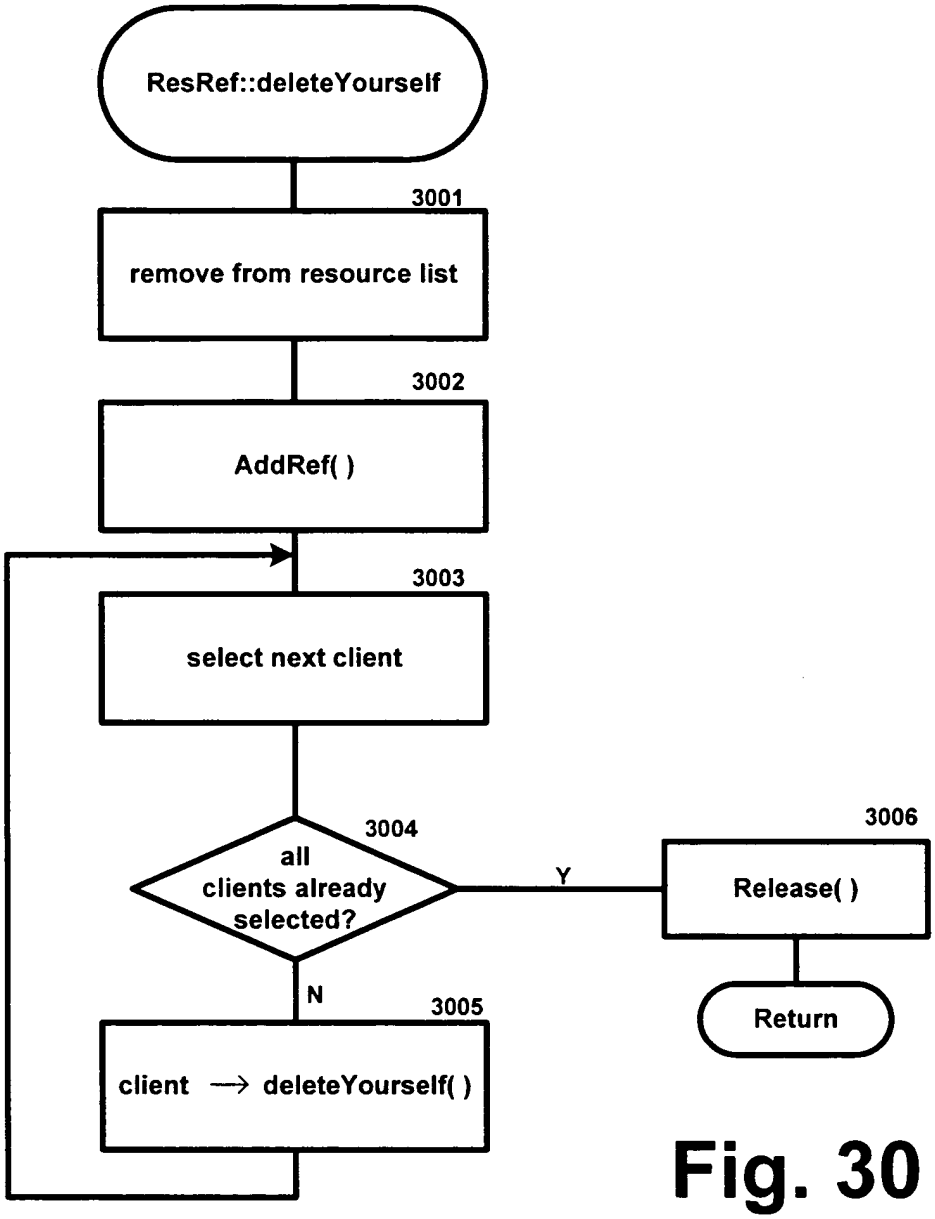

ResRef::deleteYourself
3001
remove from resource list
3002
AddRef( )
3003
select next client
3004
all clients already selected?
Y
3006
Release( )
Return
N
3005
client → deleteYourself( )

Fig. 30

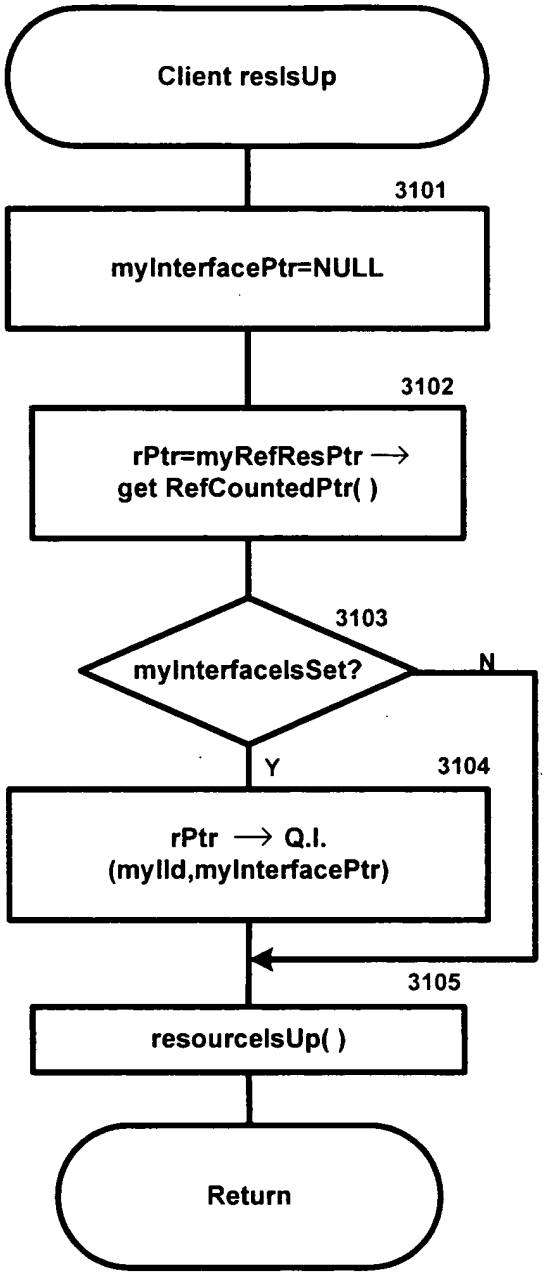

Client resIsUp
3101
myInterfacePtr=NULL
3102
rPtr=myRefResPtr →
get RefCountedPtr( )
3103
myInterfaceIsSet?
N
Y
3104
rPtr → Q.I.
(myIId,myInterfacePtr)
3105
resourceIsUp( )
Return

Fig. 31

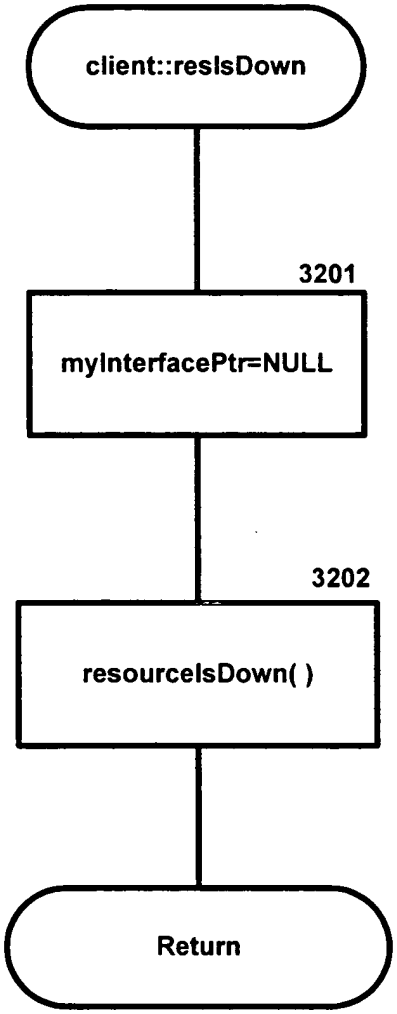

client::resIsDown
3201
myInterfacePtr=NULL
3202
resourceIsDown( )
Return

Fig. 32

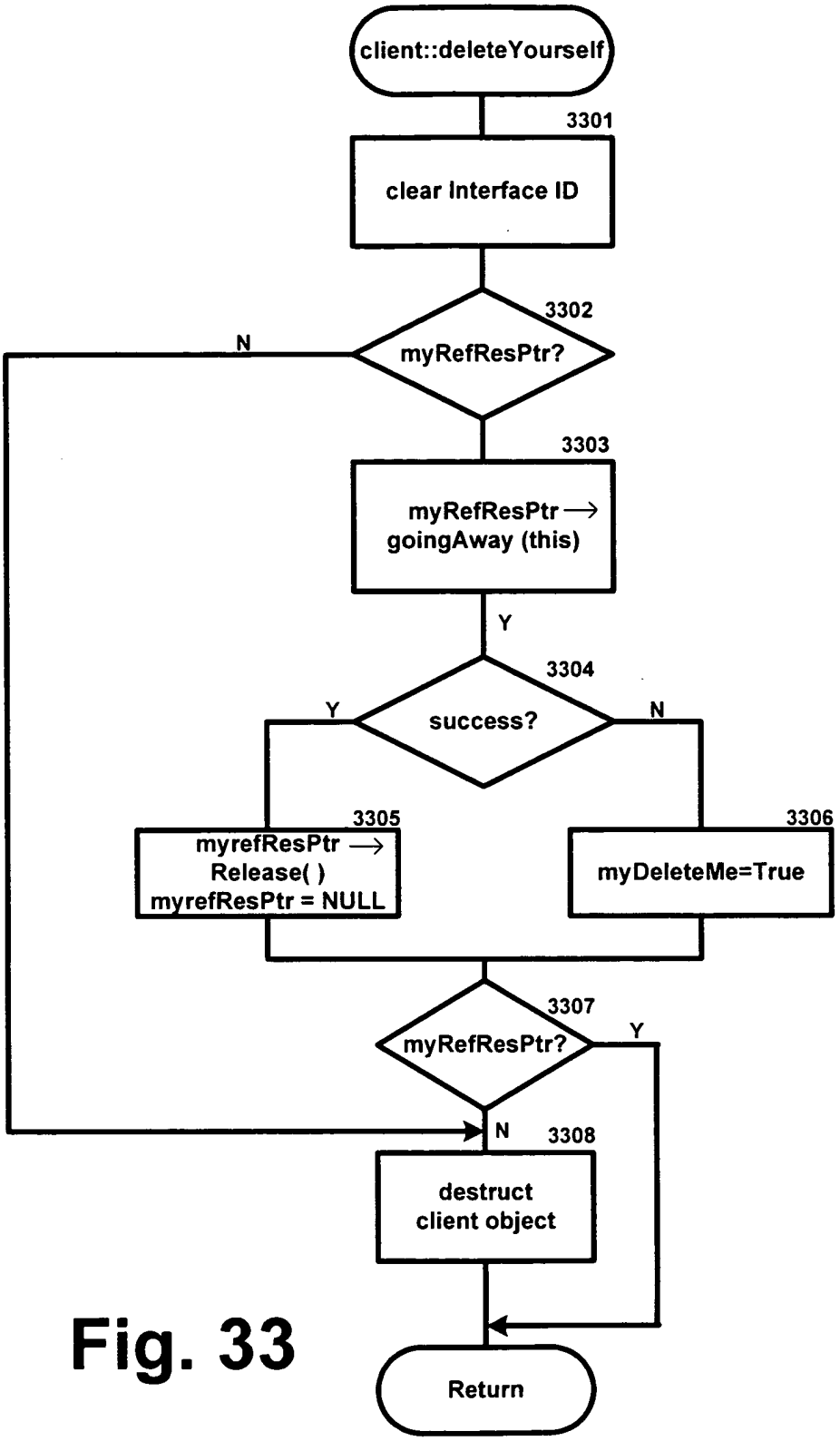

client::deleteYourself
3301
clear Interface ID
3302
myRefResPtr?
N
3303
myRefResPtr → goingAway (this)
Y
3304
success?
Y
N
3305
myrefResPtr → Release( ) myrefResPtr = NULL
3306
myDeleteMe=True
3307
myRefResPtr?
Y
N
3308
destruct client object
Return

Fig. 33

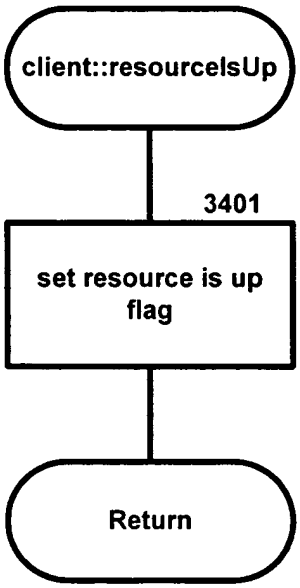


Fig. 34

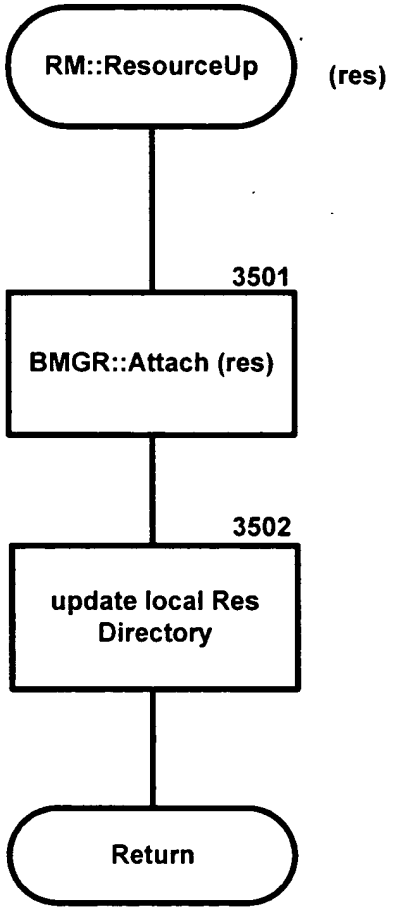

RM::ResourceUp
(res)
3501
BMGR::Attach (res)
3502
update local Res Directory
Return

Fig. 35

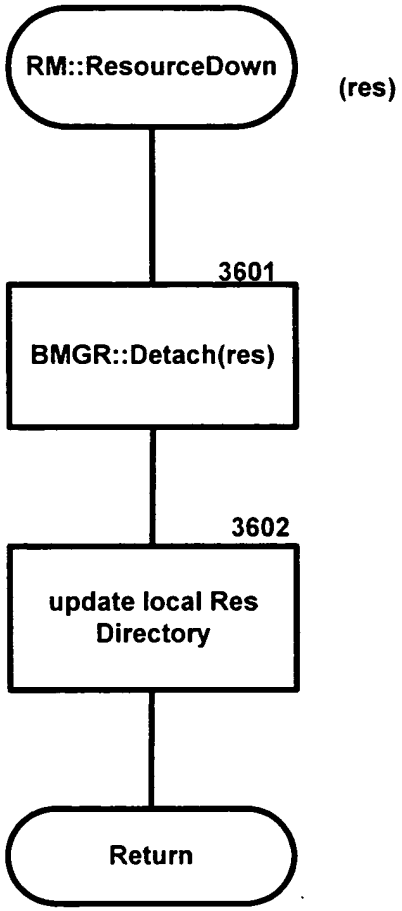

RM::ResourceDown
(res)
3601
BMGR::Detach(res)
3602
update local Res Directory
Return

Fig. 36

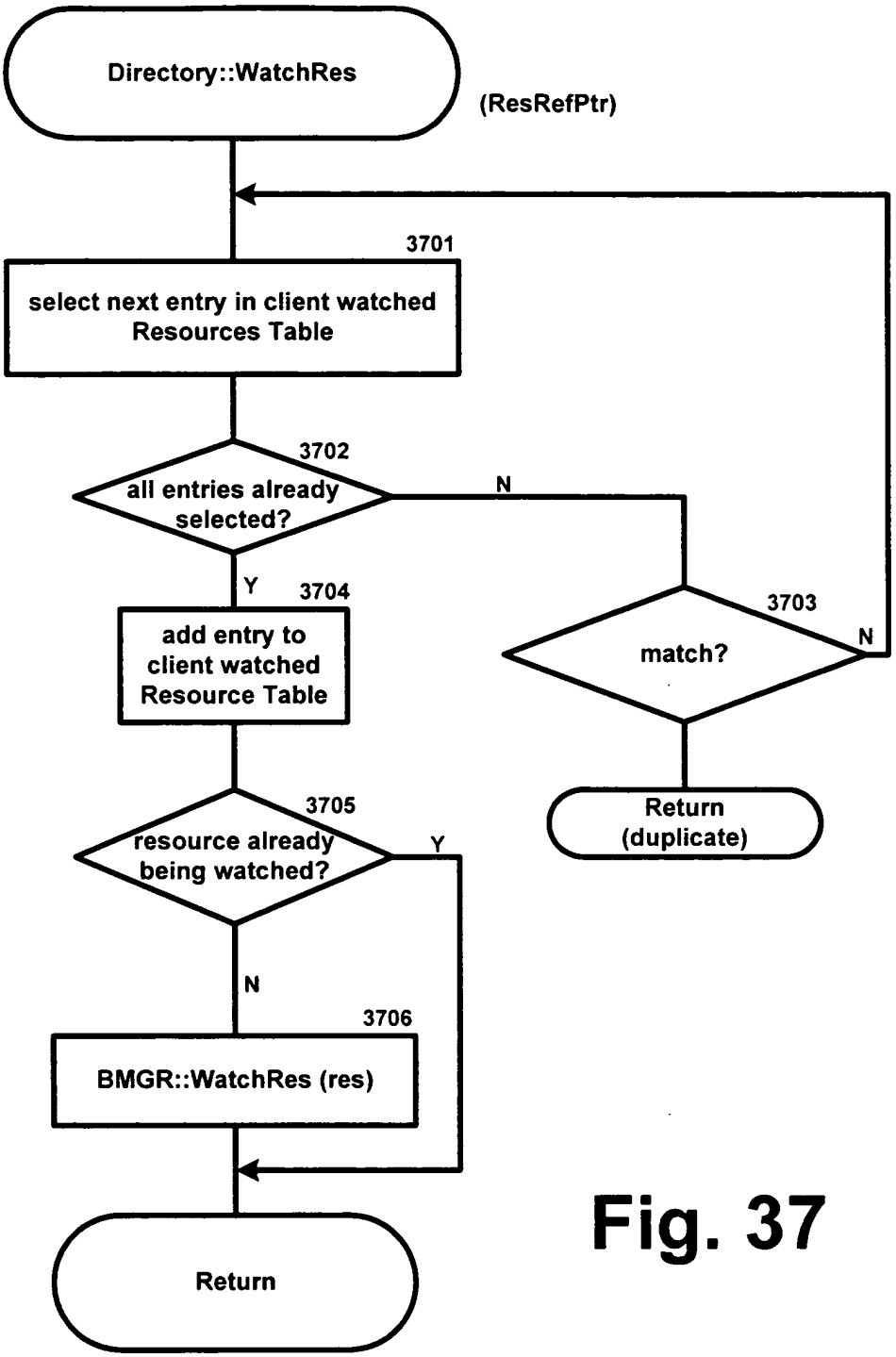

Directory::WatchRes
(ResRefPtr)
3701
select next entry in client watched Resources Table
3702
all entries already selected?
N
Y
3704
add entry to client watched Resource Table
3703
match?
N
Return (duplicate)
3705
resource already being watched?
Y
N
3706
BMGR::WatchRes (res)
Return

Fig. 37

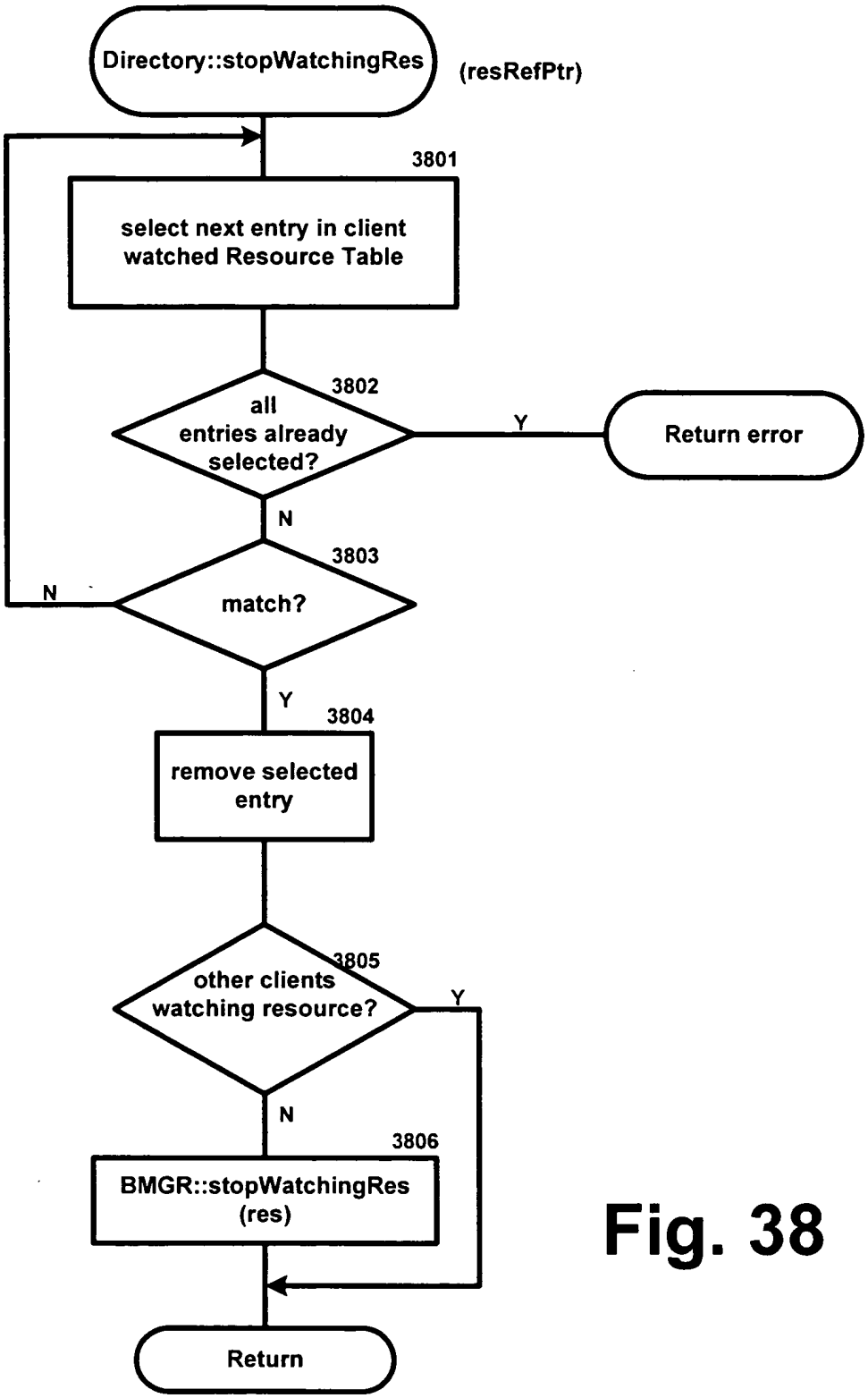

Directory::stopWatchingRes
(resRefPtr)
3801
select next entry in client watched Resource Table
3802
all entries already selected?
Y
Return error
N
3803
match?
N
Y
3804
remove selected entry
3805
other clients watching resource?
Y
N
3806
BMGR::stopWatchingRes (res)
Return

Fig. 38

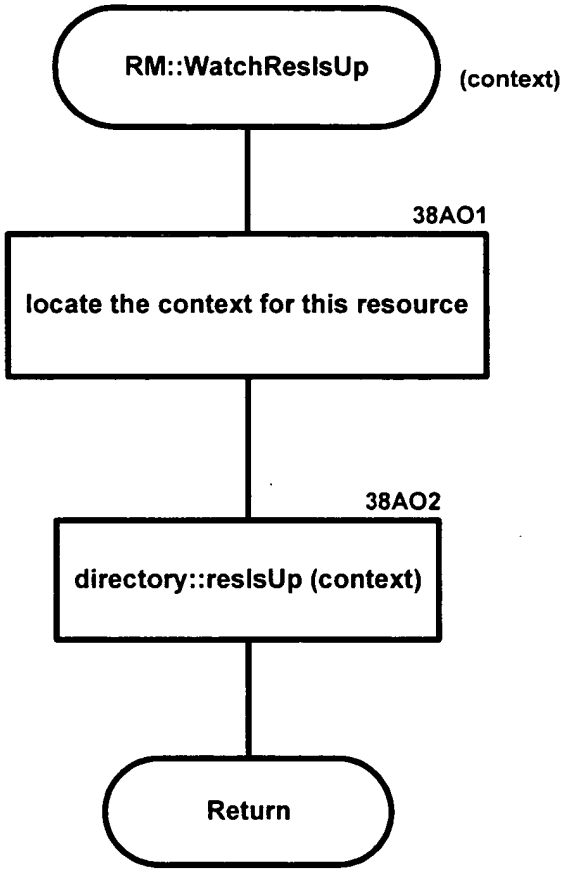

RM::WatchResIsUp
(context)
38AO1
locate the context for this resource
38AO2
directory::resIsUp (context)
Return

Fig. 38A

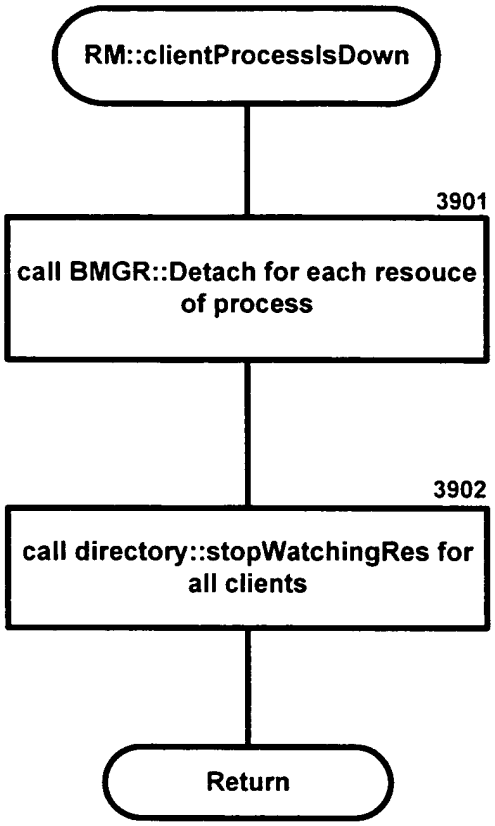

RM::clientProcessIsDown
3901
call BMGR::Detach for each resouce of process
3902
call directory::stopWatchingRes for all clients
Return

Fig. 39

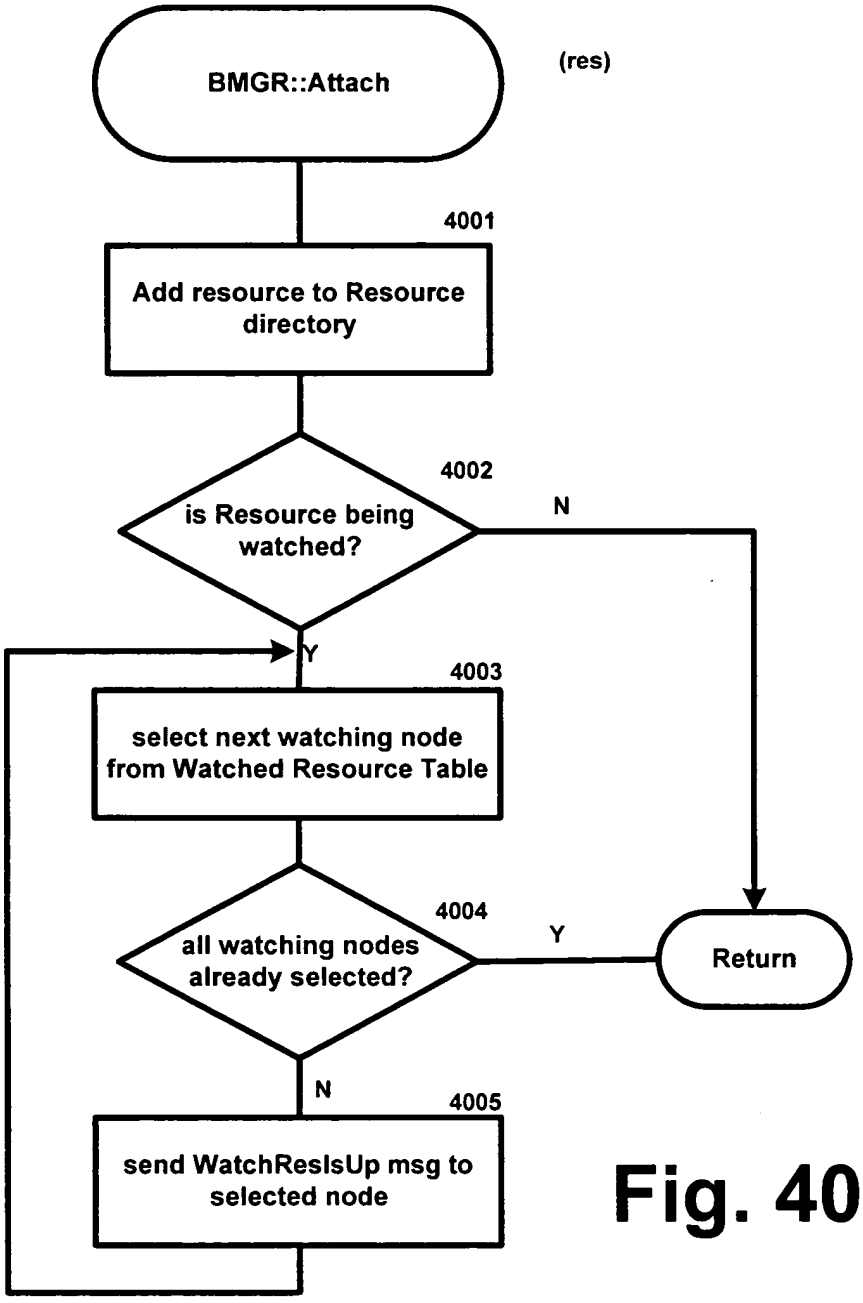

BMGR::Attach
(res)
4001
Add resource to Resource directory
4002
is Resource being watched?
N
Y
4003
select next watching node from Watched Resource Table
4004
all watching nodes already selected?
Y
Return
N
4005
send WatchResIsUp msg to selected node

Fig. 40

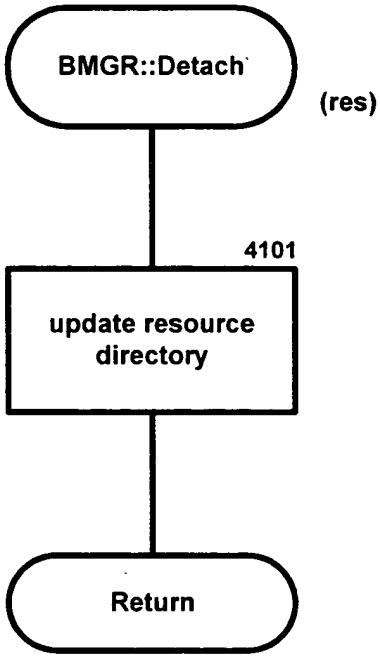


Fig. 41

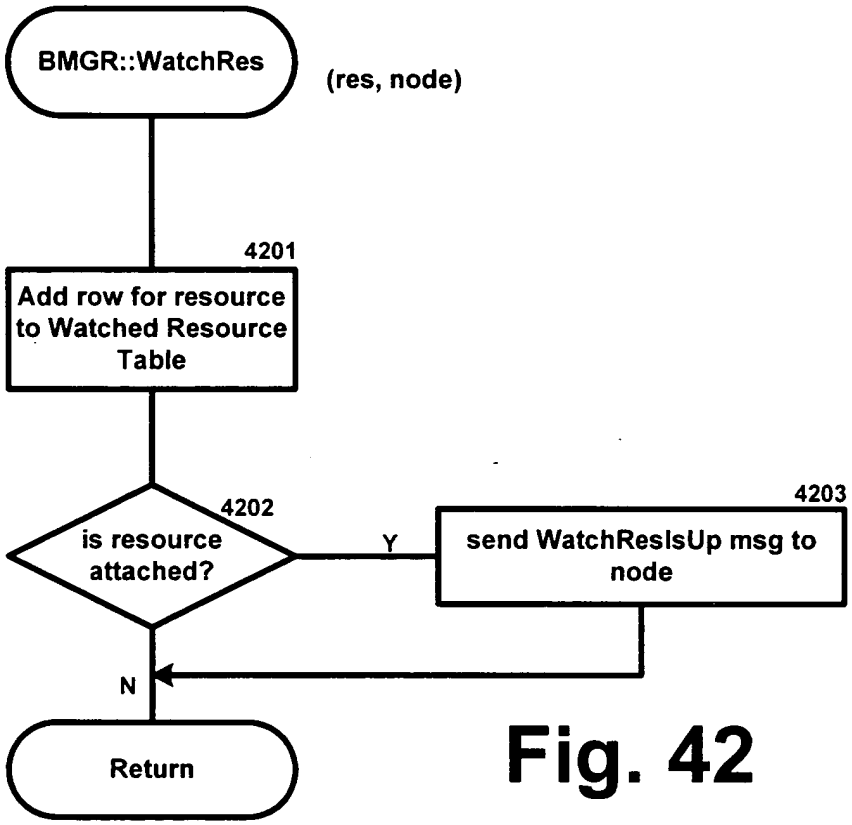

BMGR::WatchRes
(res, node)
4201
Add row for resource to Watched Resource Table
4202
is resource attached?
Y
4203
send WatchResIsUp msg to node
N
Return

Fig. 42

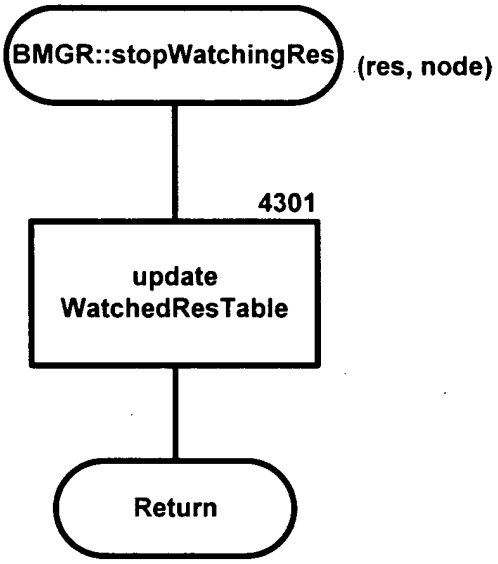

BMGR::stopWatchingRes
(res, node)
4301
update
WatchedResTable
Return

Fig. 43

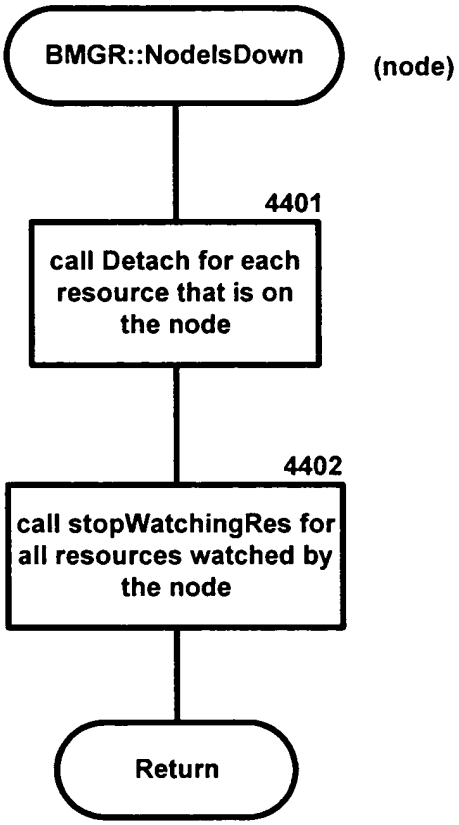

BMGR::NodeIsDown
(node)
4401
call Detach for each resource that is on the node
4402
call stopWatchingRes for all resources watched by the node
Return

Fig. 44

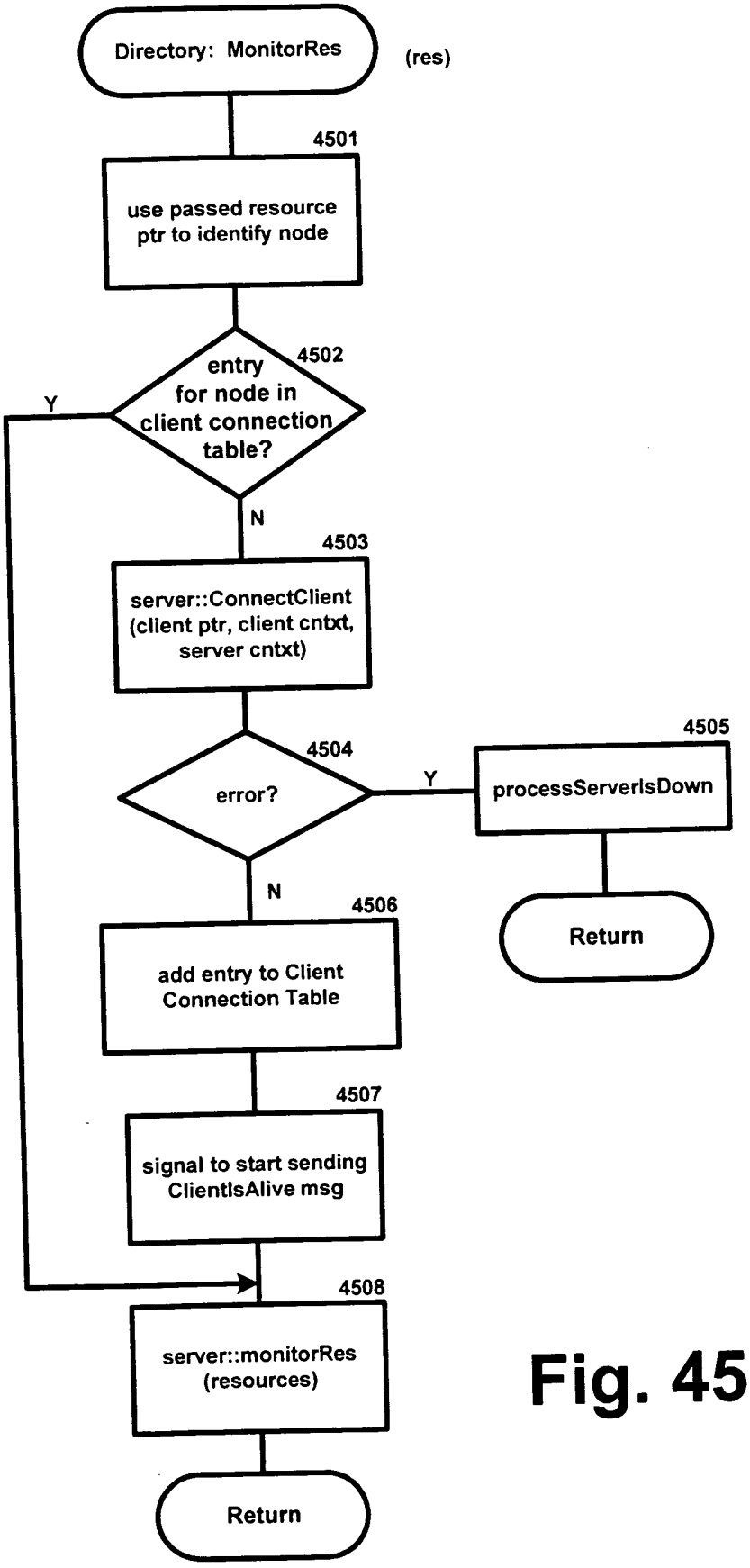

Directory: MonitorRes
(res)
4501
use passed resource ptr to identify node
4502
entry for node in client connection table?
Y
N
4503
server::ConnectClient (client ptr, client cntxt, server cntxt)
4504
error?
Y
4505
processServerIsDown
Return
N
4506
add entry to Client Connection Table
4507
signal to start sending ClientIsAlive msg
4508
server::monitorRes (resources)
Return

Fig. 45

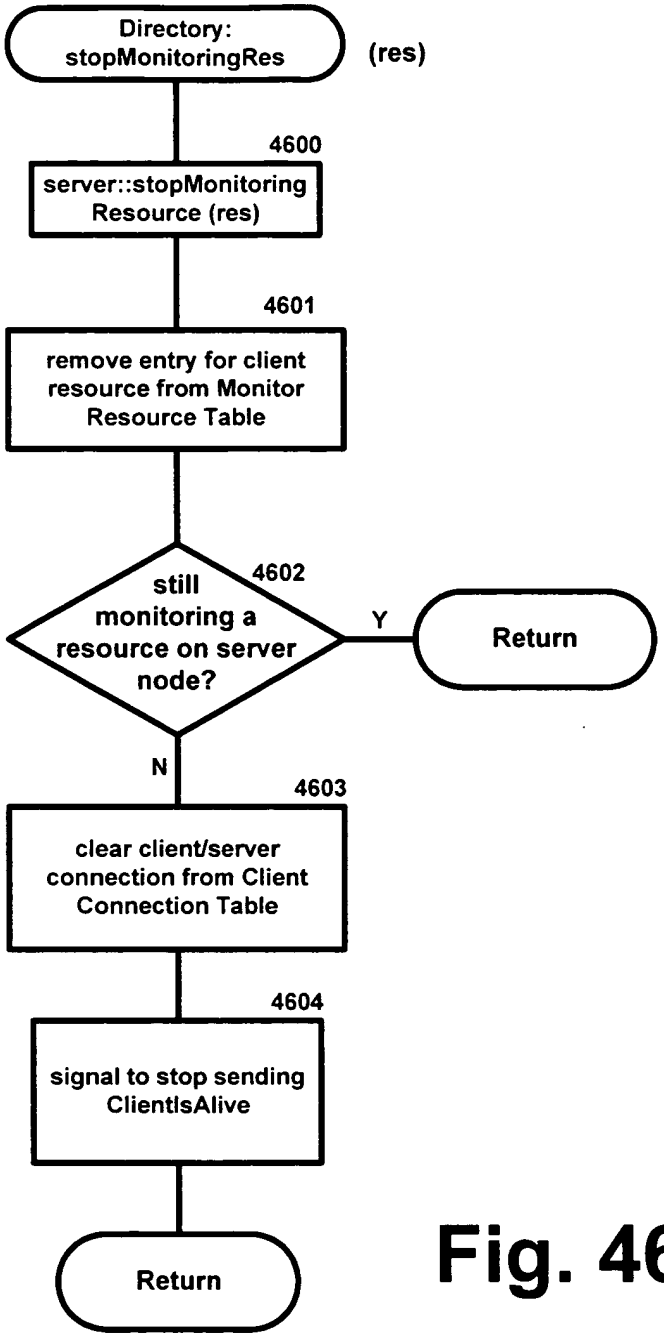

Directory: stopMonitoringRes
(res)
4600
server::stopMonitoring Resource (res)
4601
remove entry for client resource from Monitor Resource Table
4602
still monitoring a resource on server node?
Y
Return
N
4603
clear client/server connection from Client Connection Table
4604
signal to stop sending ClientIsAlive
Return

Fig. 46

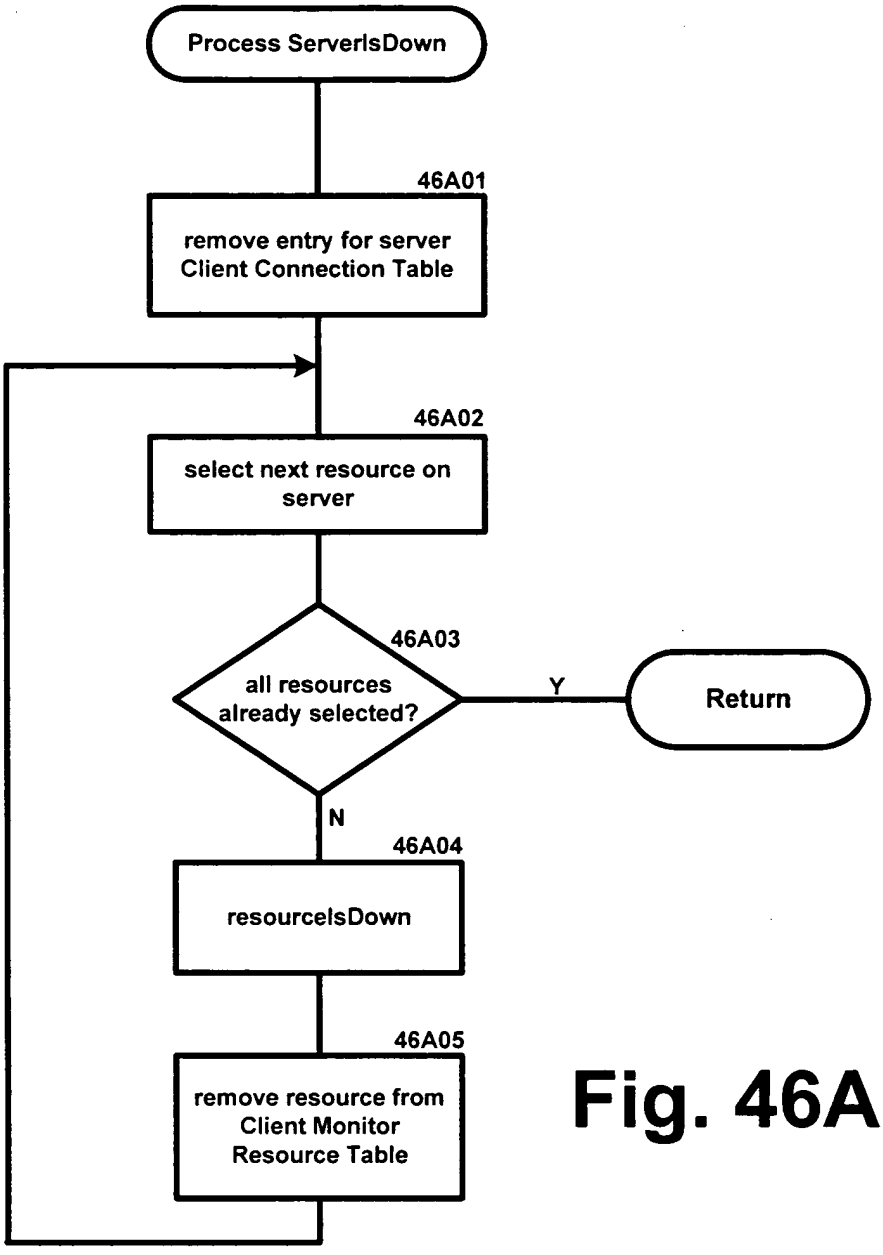

Process ServerIsDown
46A01
remove entry for server Client Connection Table
46A02
select next resource on server
46A03
all resources already selected?
Y
Return
N
46A04
resourceIsDown
46A05
remove resource from Client Monitor Resource Table

Fig. 46A

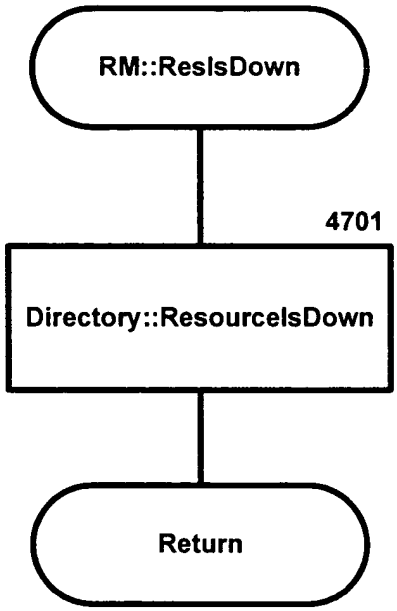


Fig. 47

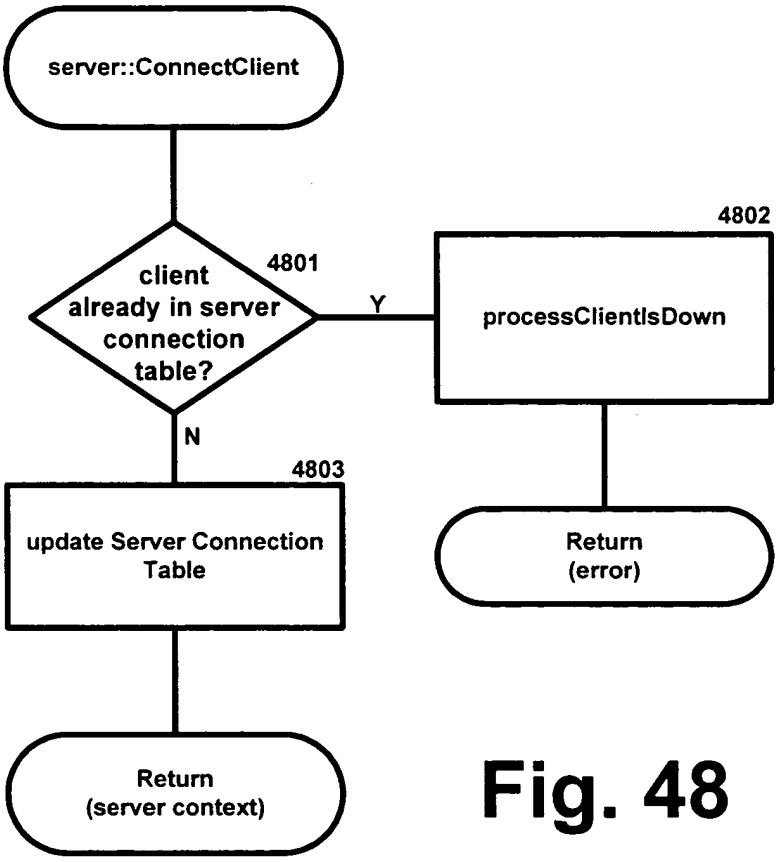

server::ConnectClient
client already in server connection table?
4801
Y
N
4802
processClientIsDown
4803
update Server Connection Table
Return (error)
Return (server context)

Fig. 48

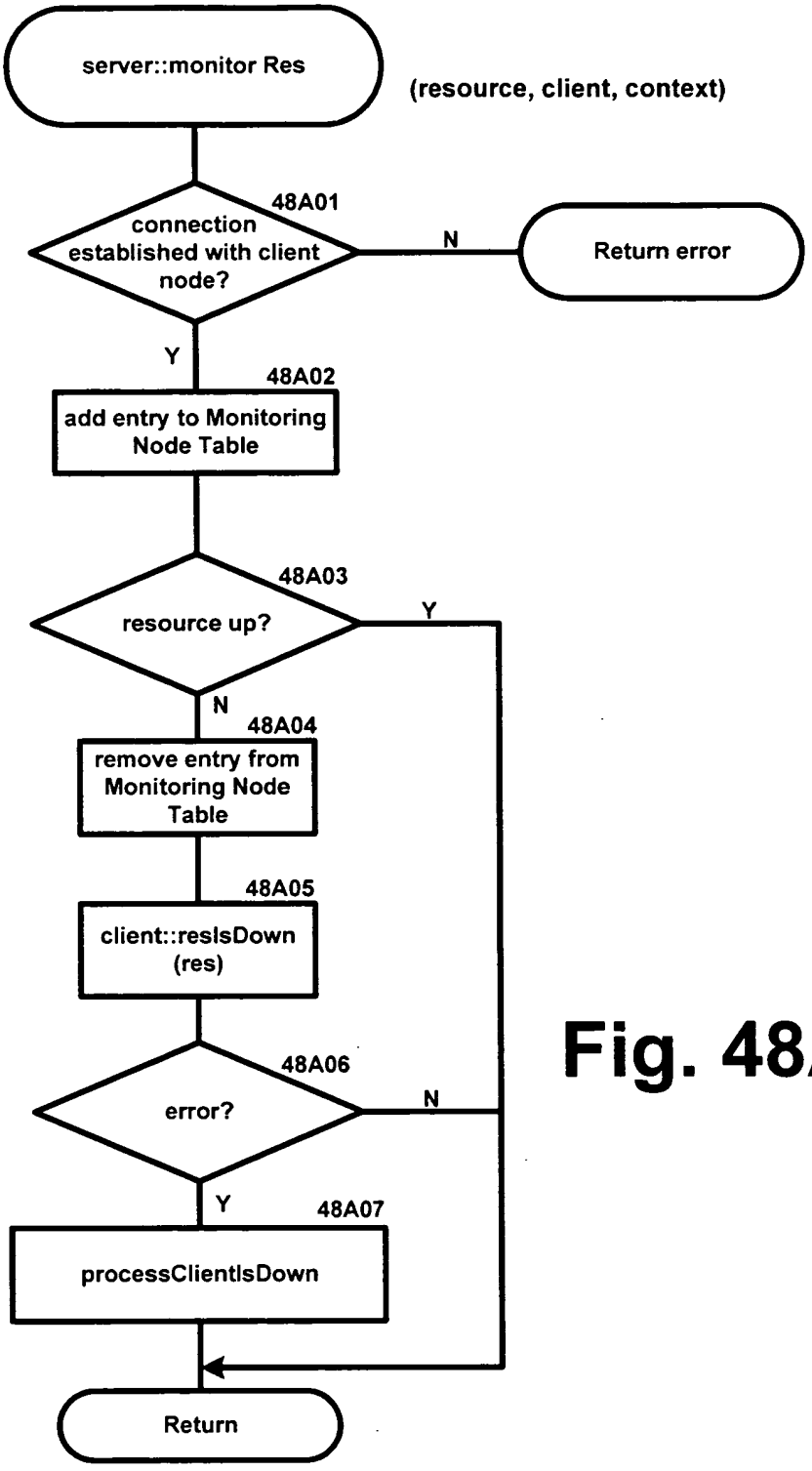

server::monitor Res
(resource, client, context)
48A01
connection established with client node?
N
Return error
Y
48A02
add entry to Monitoring Node Table
48A03
resource up?
Y
N
48A04
remove entry from Monitoring Node Table
48A05
client::resIsDown (res)
48A06
error?
N
Y
48A07
processClientIsDown
Return

Fig. 48A

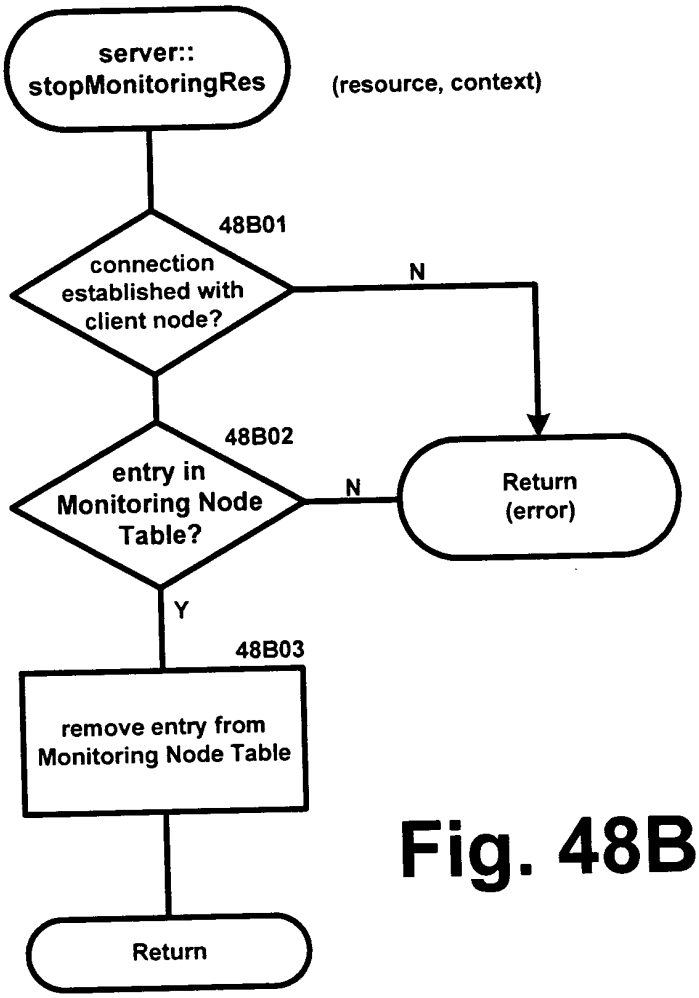

server::
stopMonitoringRes
(resource, context)
48B01
connection established with client node?
N
48B02
entry in Monitoring Node Table?
N
Return (error)
Y
48B03
remove entry from Monitoring Node Table
Return

Fig. 48B

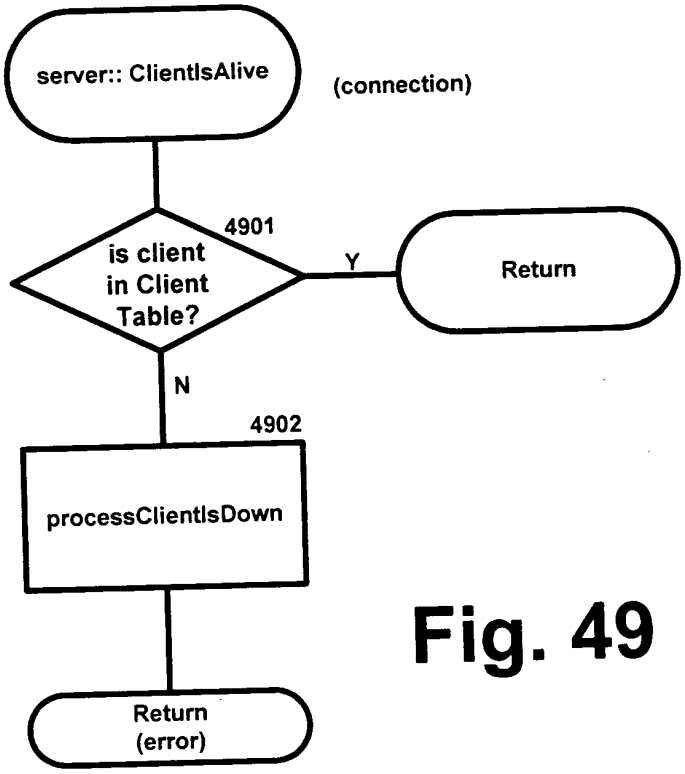

server:: ClientIsAlive
(connection)
4901
is client in Client Table?
Y
Return
N
4902
processClientIsDown
Return (error)

Fig. 49

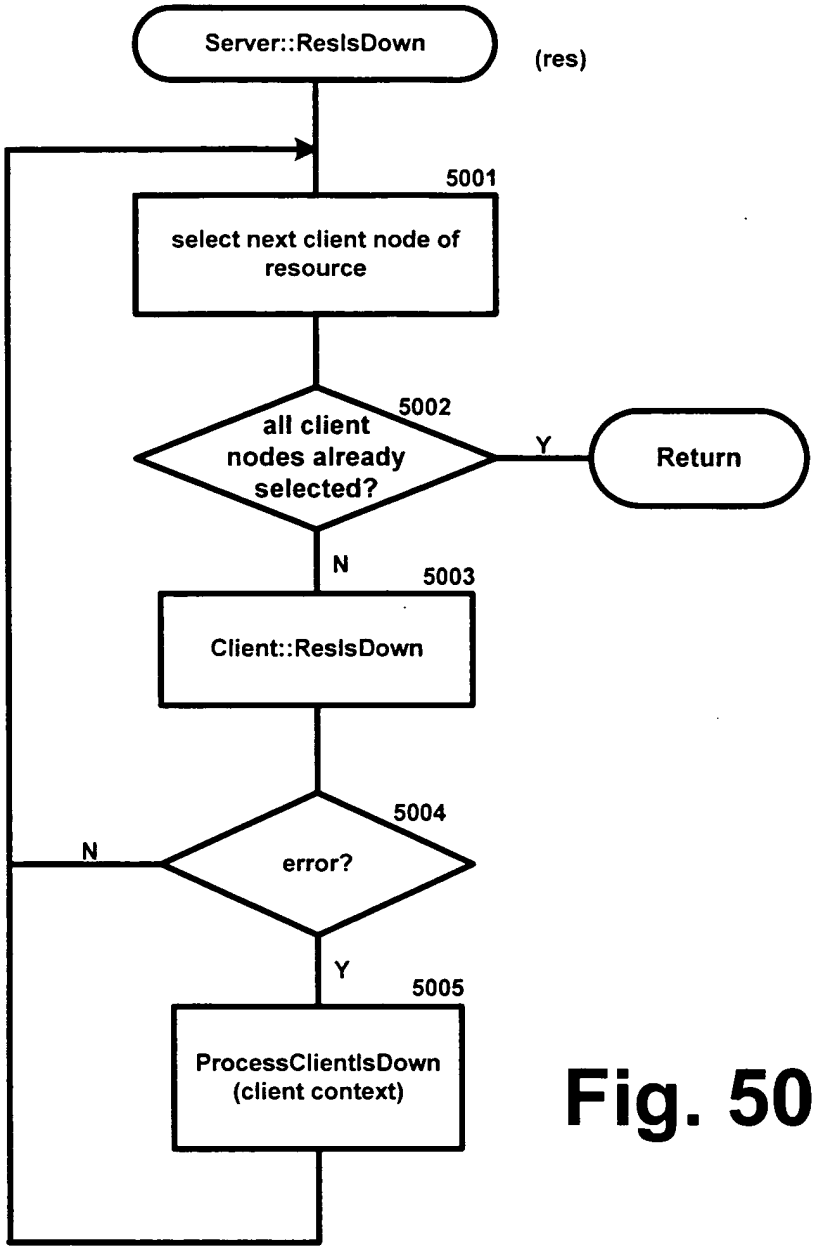

Server::ResIsDown
(res)
5001
select next client node of resource
5002
all client nodes already selected?
Y
Return
N
5003
Client::ResIsDown
5004
error?
N
Y
5005
ProcessClientIsDown (client context)

Fig. 50

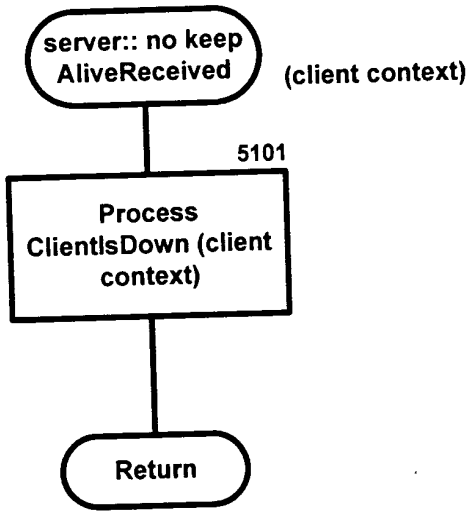

server:: no keep AliveReceived
(client context)
5101
Process ClientIsDown (client context)
Return

Fig. 51

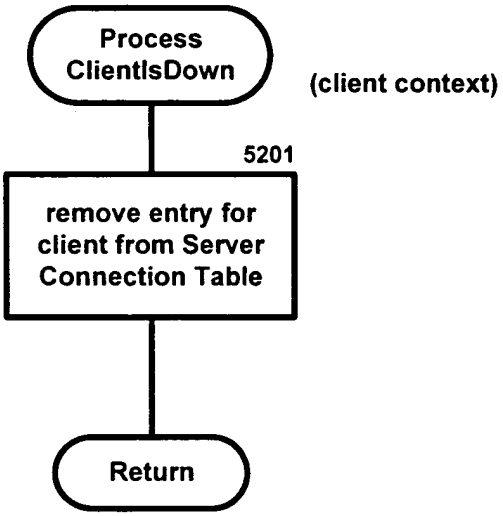
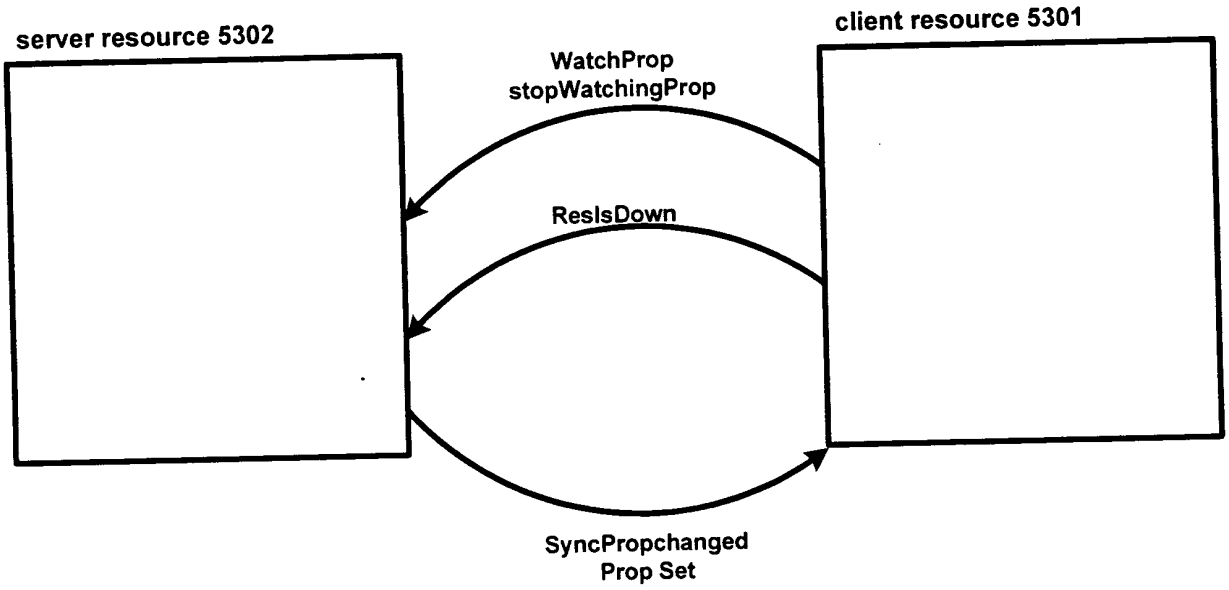

Process ClientIsDown
(client context)
5201
remove entry for client from Server Connection Table
Return

Fig. 52

Fig. 53


server resource 5302
client resource 5301
WatchProp
stopWatchingProp
ResIsDown
SyncPropchanged
Prop Set

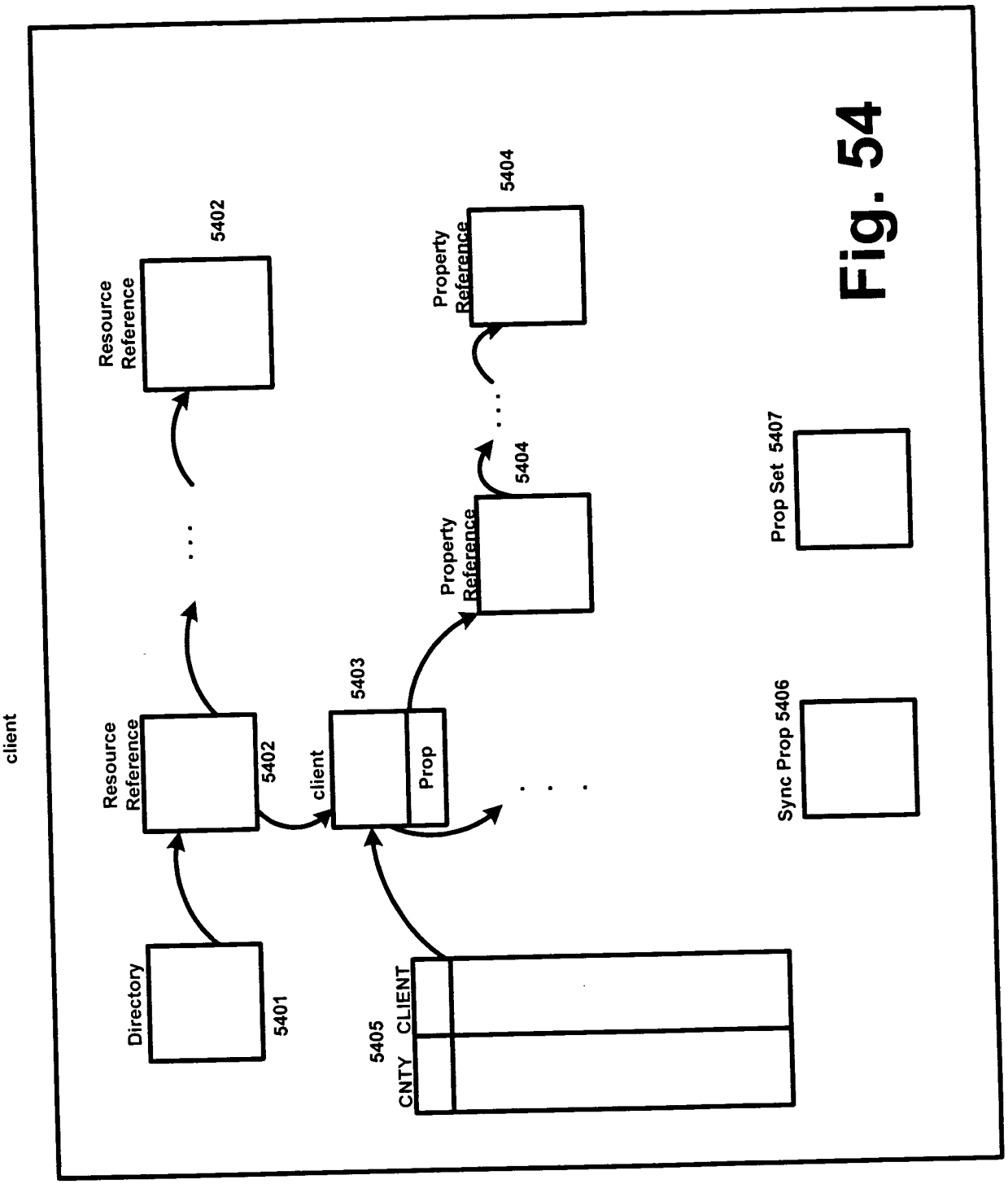
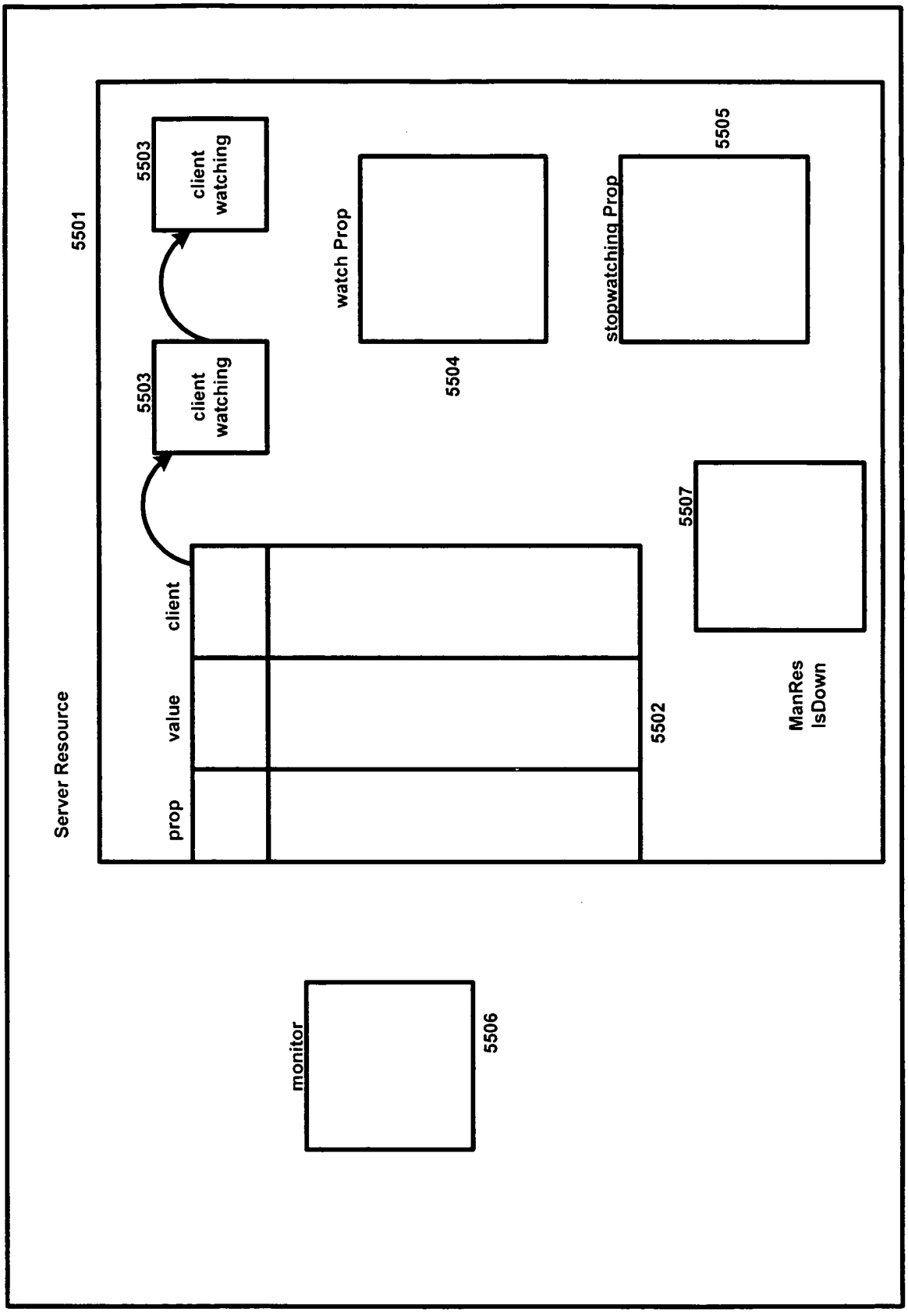

client
Directory
5401
5405
CNTY
CLIENT
Resource Reference
5402
client
5403
Prop
Property Reference
5404
Resource Reference
5402
Property Reference
5404
Prop Set 5407
Sync Prop 5406

Fig. 54

Fig. 55


Server Node
Server Resource
5501
client
value
prop
5502
5503
client watching
5503
client watching
watch Prop
5504
stopwatching Prop
5505
5507
ManRes
IsDown
monitor
5506

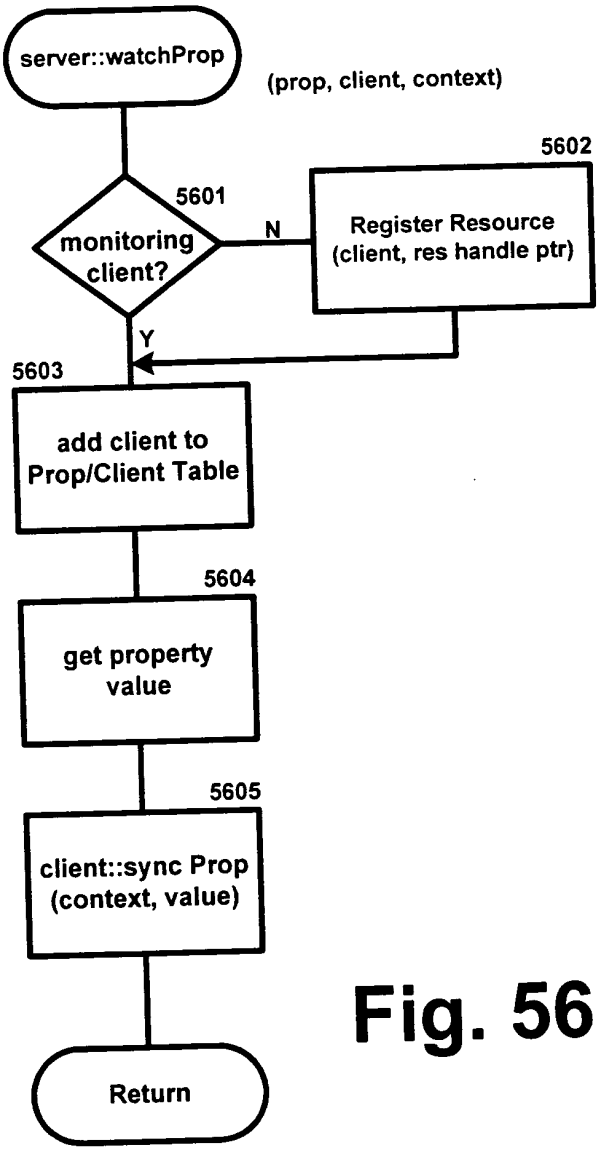

server::watchProp
(prop, client, context)
5601
monitoring client?
N
5602
Register Resource (client, res handle ptr)
Y
5603
add client to Prop/Client Table
5604
get property value
5605
client::sync Prop (context, value)
Return

Fig. 56

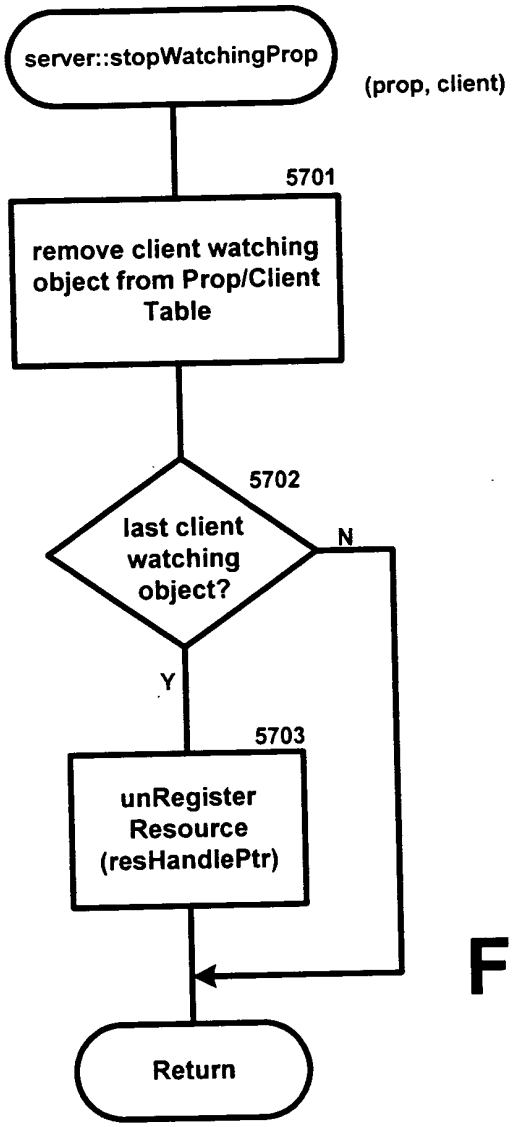

server::stopWatchingProp
(prop, client)
5701
remove client watching object from Prop/Client Table
5702
last client watching object?
N
Y
5703
unRegister Resource (resHandlePtr)
Return

Fig. 57

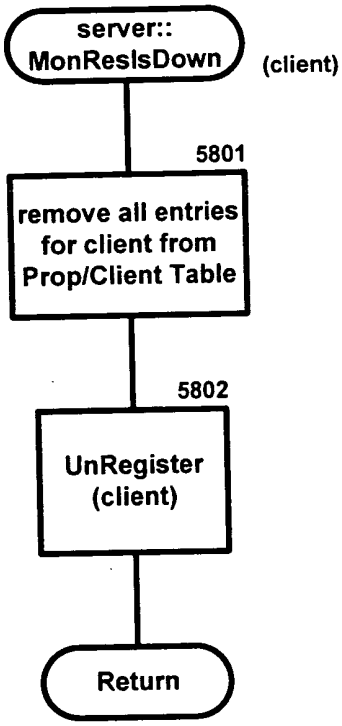

server::
MonResIsDown
(client)
5801
remove all entries for client from Prop/Client Table
5802
UnRegister (client)
Return

Fig. 58

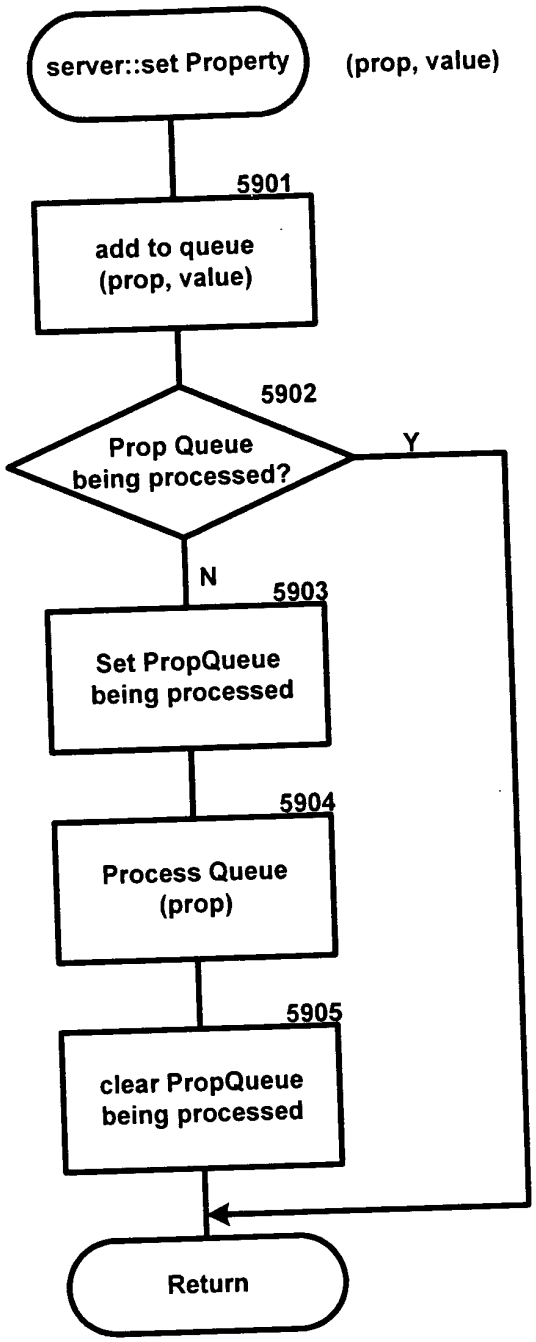

server::set Property
(prop, value)
5901
add to queue
(prop, value)
5902
Prop Queue
being processed?
Y
N
5903
Set PropQueue
being processed
5904
Process Queue
(prop)
5905
clear PropQueue
being processed
Return

Fig. 59

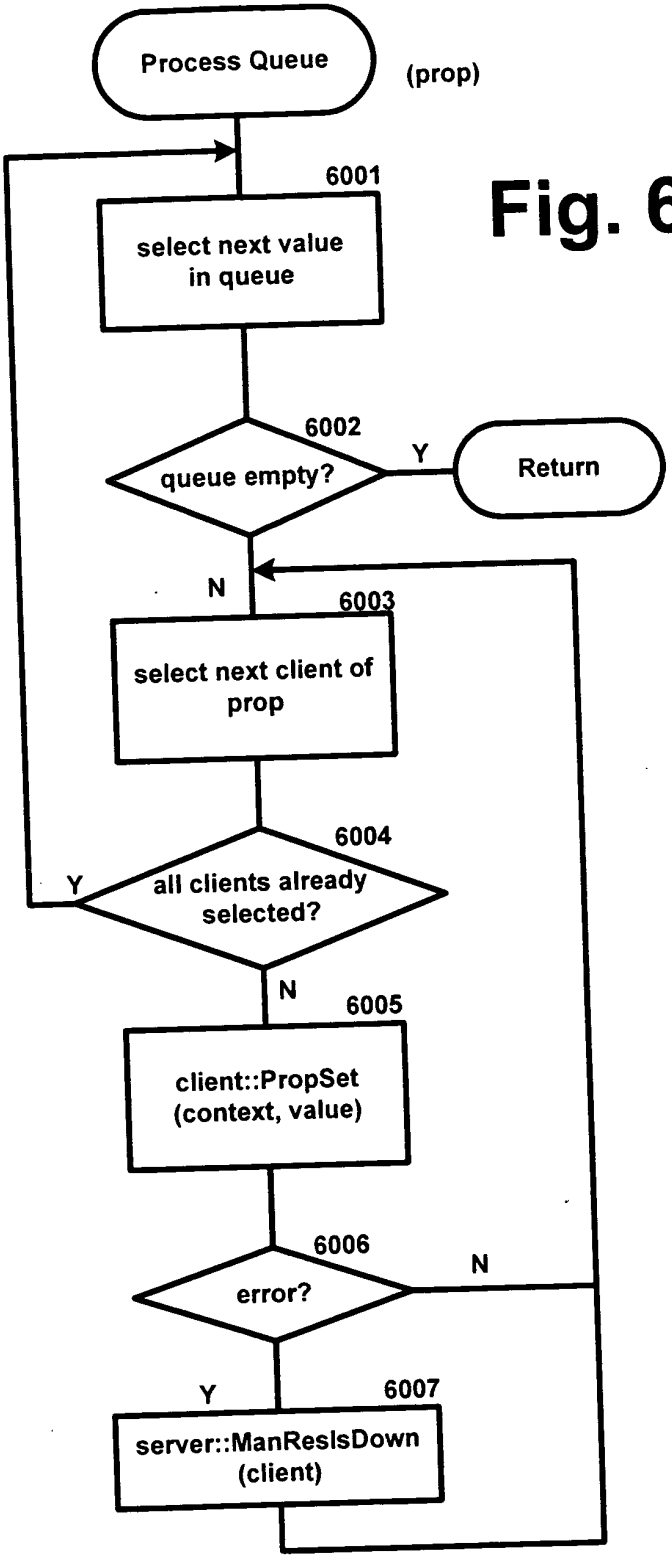

Process Queue
(prop)
6001
select next value in queue
6002
queue empty?
Y
Return
N
6003
select next client of prop
6004
all clients already selected?
Y
N
6005
client::PropSet (context, value)
6006
error?
N
Y
6007
server::ManResIsDown (client)

Fig. 60

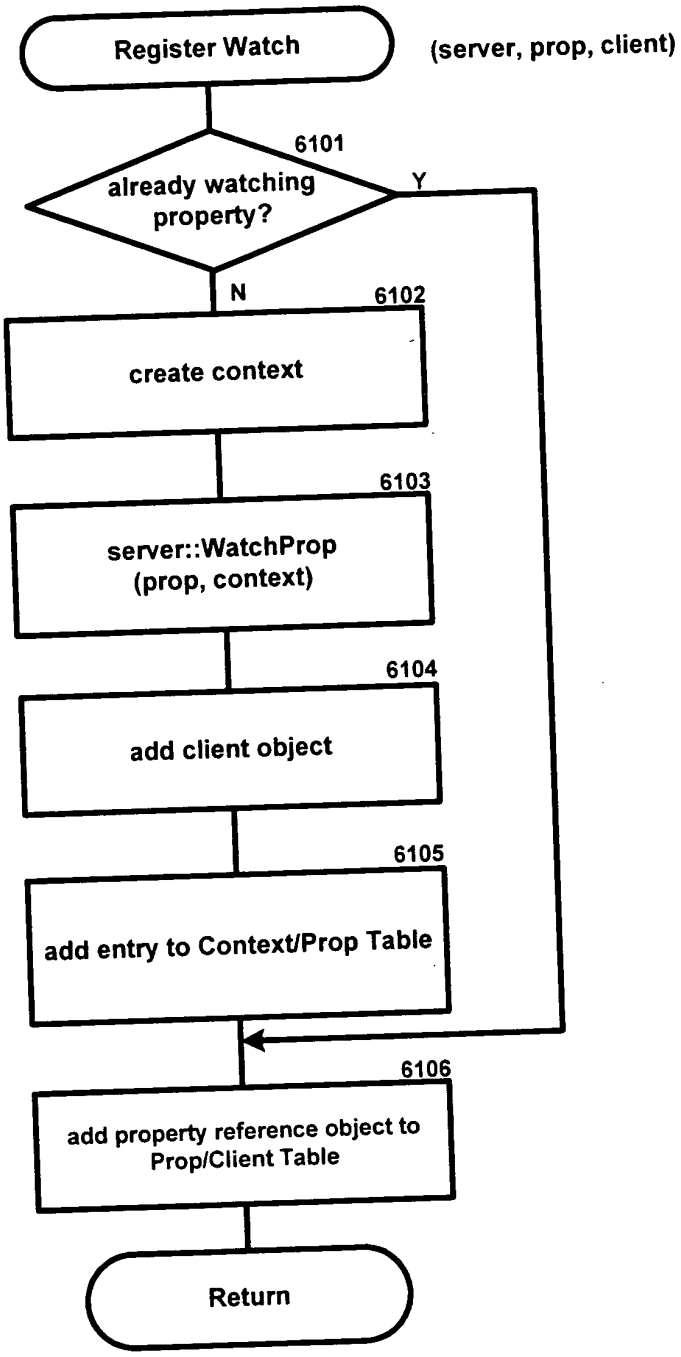

Register Watch
(server, prop, client)
6101
already watching property?
Y
N
6102
create context
6103
server::WatchProp (prop, context)
6104
add client object
6105
add entry to Context/Prop Table
6106
add property reference object to Prop/Client Table
Return

Fig. 61

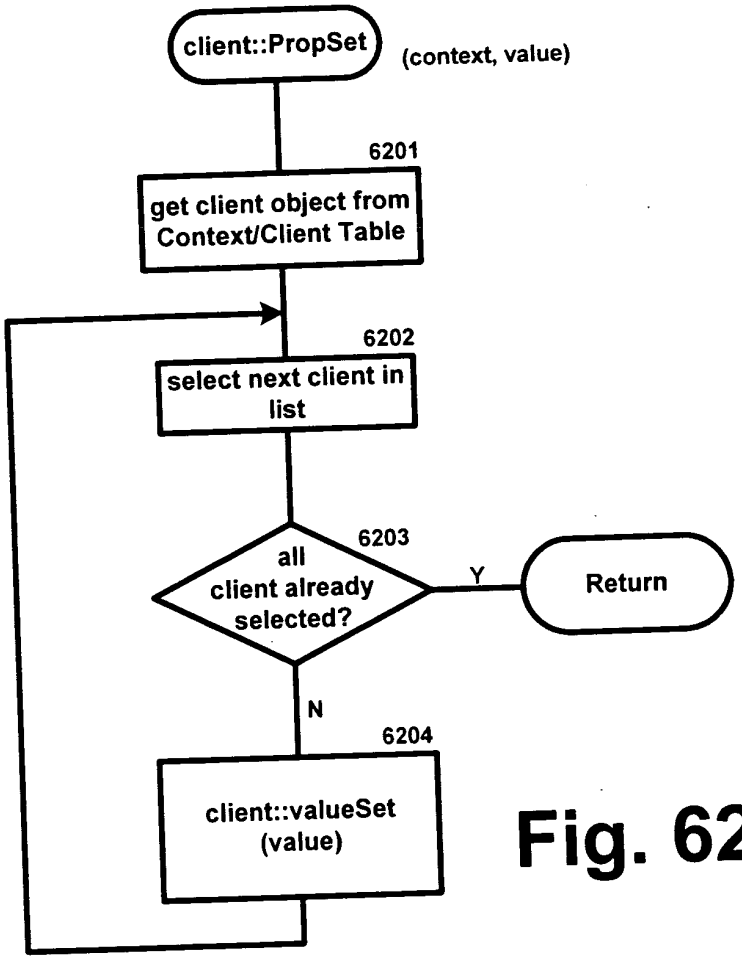

client::PropSet
(context, value)
6201
get client object from Context/Client Table
6202
select next client in list
6203
all client already selected?
Y
Return
N
6204
client::valueSet (value)

Fig. 62

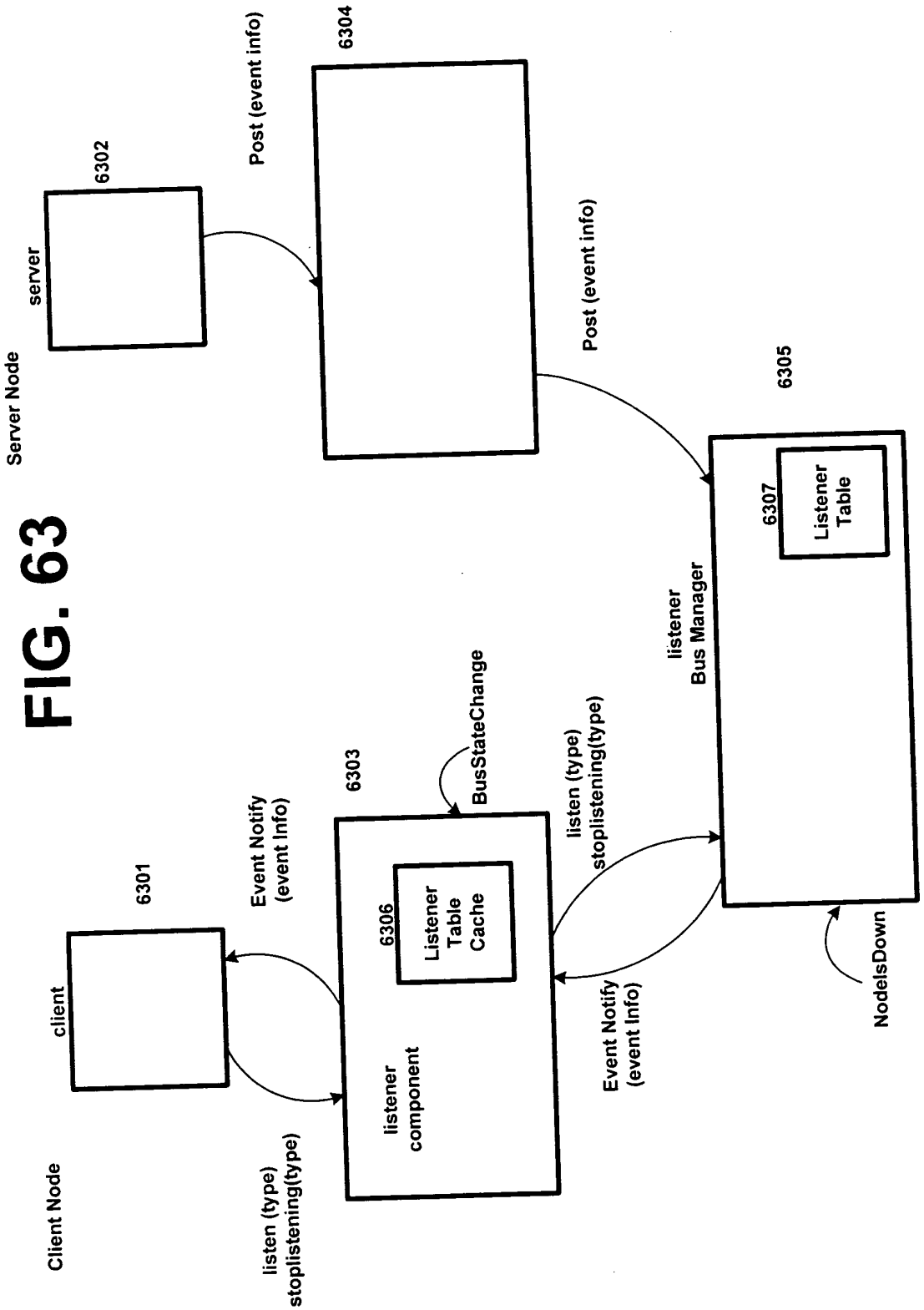

Server Node
server
6302
Post (event info)
6304
Post (event info)
6305
6307
Listener Table
listener Bus Manager
BusStateChange
6303
listen (type) stoplistening(type)
Event Notify (event Info)
6301
6306
Listener Table Cache
client
listener component
Event Notify (event Info)
NodeIsDown
Client Node
listen (type) stoplistening(type)

FIG. 63